Lecture 10: More Advanced ER Modeling Techniques and Design Alternatives

CSX3006 DATABASE SYSTEMS

ITX3006 DATABASE MANAGEMENT SYSTEMS

Outline

- Extended ER Constructs
  - Strong Entity vs. Weak Entity
  - Inheritance Hierarchy: Specialization / Generalization
  - Aggregation
- Design Alternatives
  - Simple Attribute vs. Composite Attribute
  - Entity Sets vs. Attributes
  - Entity Sets vs. Relationship Sets
  - Attributes vs. Relationship Sets
  - Placement of Descriptive Attributes (Attributes of a Relationship Set)
  - Weak Entity Sets vs. Strong Entity Sets
  - Weak Entity Sets vs. Multi-valued Attributes
  - Generalization/Specialization Hierarchy vs. Separate Entities
  - N-ary relationships vs. multiple binary relationships

Strong Entity Set vs Weak Entity Set

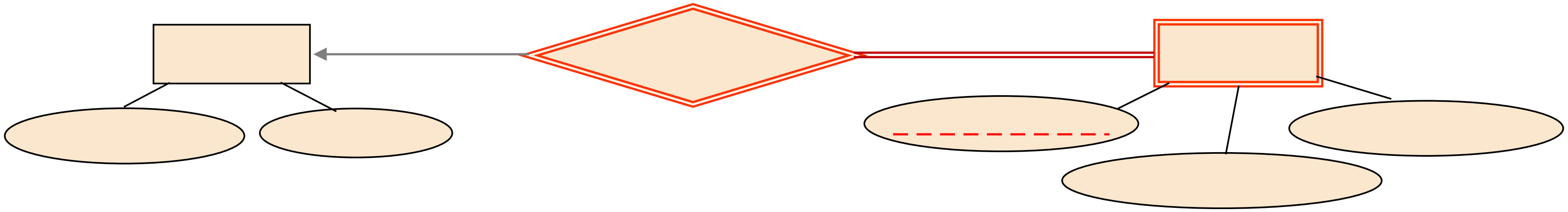
STRONG ENTITY SET

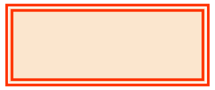
- An entity set that can **exist independently** of other entity sets
  - It has a unique characteristic (*an **identifier or key***)— an attribute or a combination of attributes that uniquely distinguish each occurrence of that entity.
  - E.g., Entity Sets: **Customer**, **Loan**, **Branch**

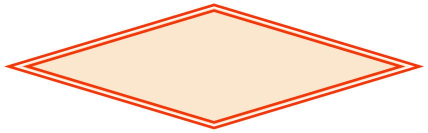
WEAK ENTITY SET

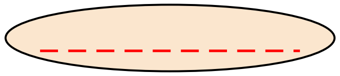
- An entity set whose **existence depends on** some **other** entity sets
  - It has an attribute that serves as a ***partial identifier (or discriminator)***.
  - The associated entity set is called *Owner Entity Set (or Identifying Entity Set)*
- E.g., Entity Set: **Payment** with attributes *payment_number, payment_timestamp, payment_amount*

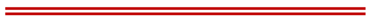
Representing Weak Entity Set and Identifying Relationship in ER Diagram - 1



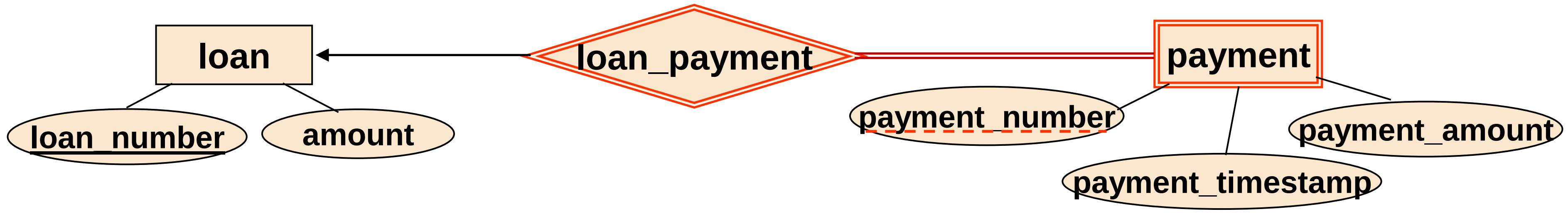
 : **Weak Entity Set**

 : **Identifying Relationship**

 : **Discriminator (attribute)**

 : **Total participation of the Weak Entity Set**

Representing Weak Entity Set and Identifying Relationship in ER Diagram - 2



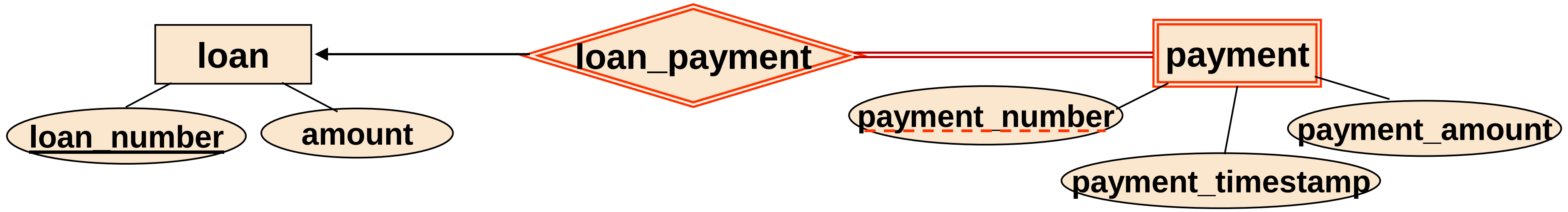
A weak entity MUST be associated with an identifying entity (owner entity) through an identifying relationship.

- loan is an identifying (owner) entity set of payment
- loan_payment is the identifying relationship set

The identifying relationship

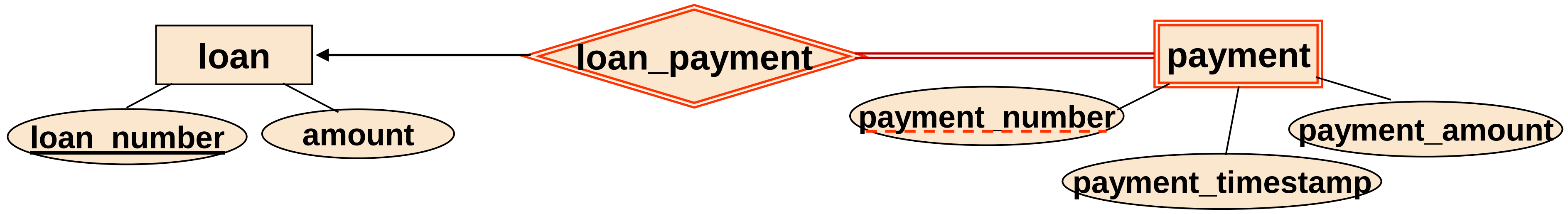
1. **Many-to-one from the weak entity set to the identifying entity set, and**
2. **The participation of the weak entity set in the relationship is total.**

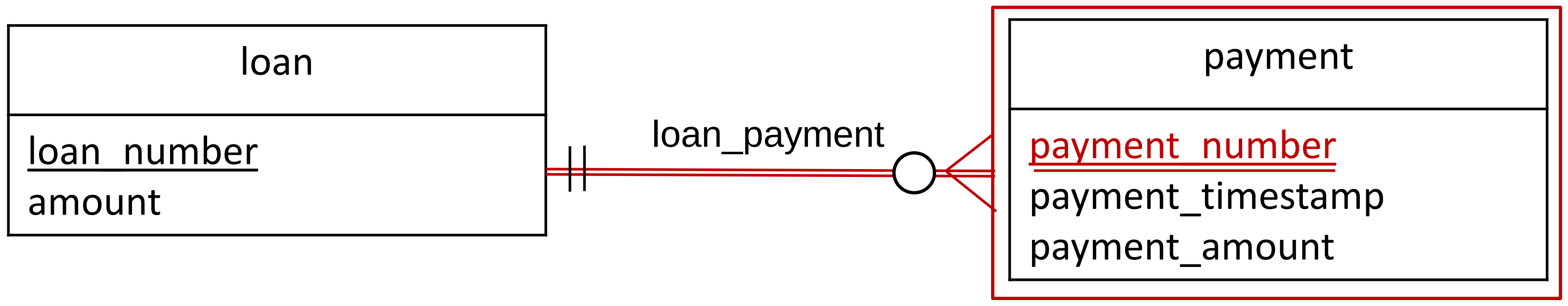
Weak Entity Set Example 1



- payment_number is a sequential number, starting from 1, generated **for each loan**
  - each payment entity is distinct with regard to each loan entity
- The **arrow** from *loan_payment* to *loan* indicates that each payment is for a single loan.

Alternative Notation

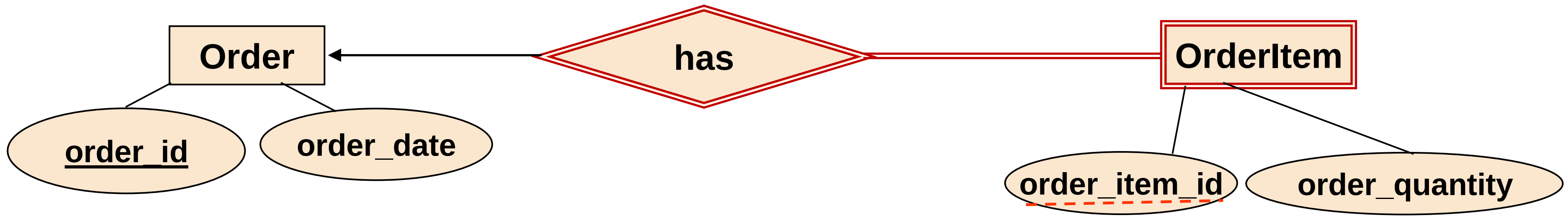




Creating a Primary Key of Weak Entity Set (Later on)

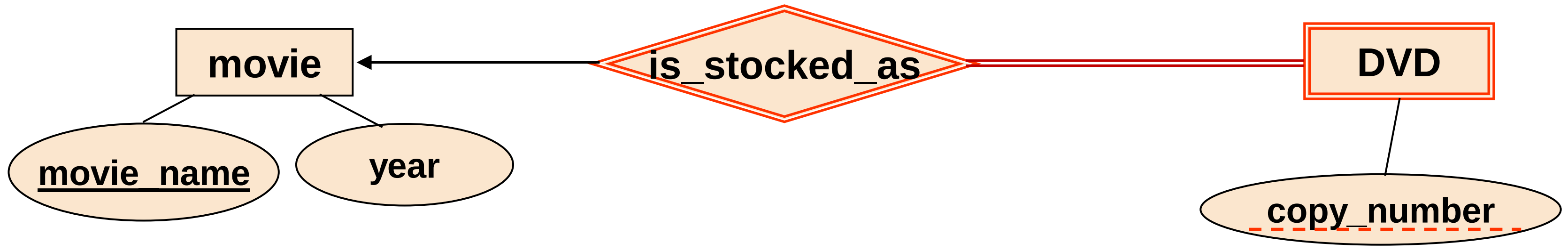
- Way 1: primary key of the identifying entity set **+** discriminator of the weak entity set
  - The discriminator (partial identifier): attribute(s) in the weak entity set to distinguishing among all weak entities
- Way 2: Use surrogate key (surrogate identifier attribute)
  - Generated by the system

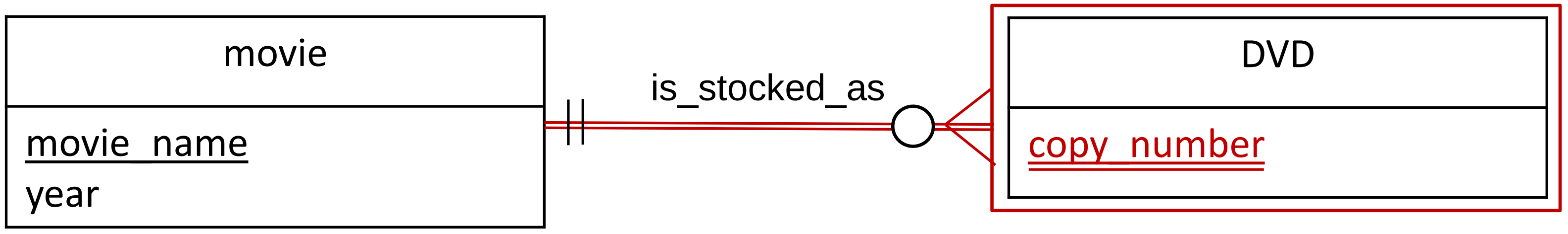
Weak Entity Set Example 2:
A database which keeps track of customer orders



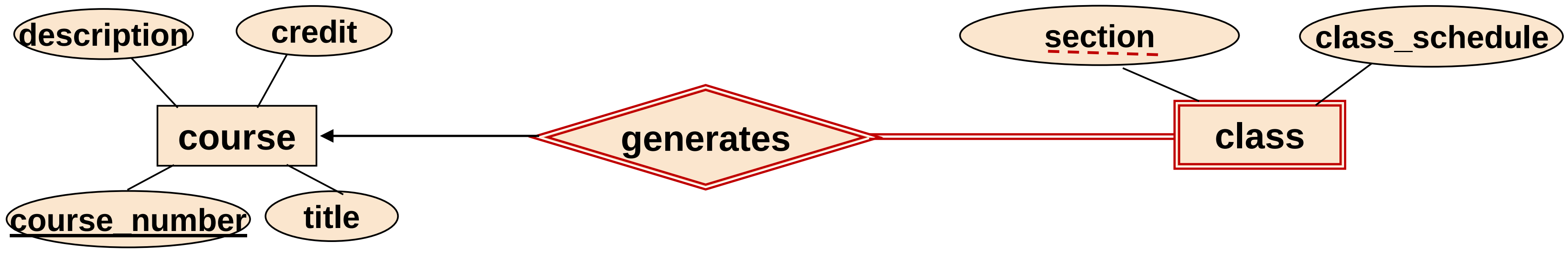
- A customer makes an order
- An *order* consists of one or more items that the enterprise sells
  - An *order* consists of one or more *orderItems*
  - An *orderItem* belongs to *a product/service bought*
- An *orderitem cannot exists without* the corresponding order
  - An *orderitem* is a weak entity whose existence depends on an identifying entity (*order* entity)

Weak Entity Set Example 3

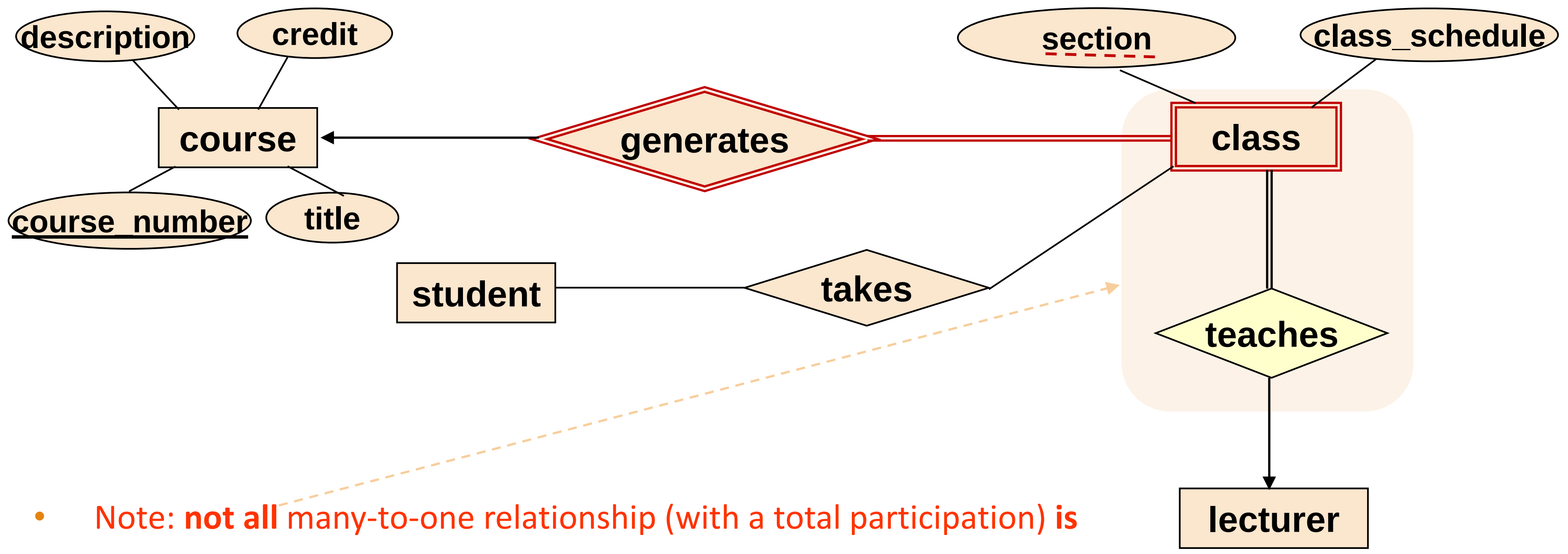




Weak Entity Set Example 4



A weak entity set *can participate* in relationships *other than the identifying relationship*



- Note: **not all** many-to-one relationship (with a total participation) **is** an identifying relationship.
  - e.g., class is not a weak entity <u>of a lecturer</u>, even though
    - 1) *teaches* is ***many-to-one*** *from class to lecturer,* and
    - 2) *class is having a* ***total participation***
    - A particular class can be identified from a *course number* and *section* number without requiring a lecturer entity

When to Adapt Weak Entity set?

- ER Modelling is a means of **capturing semantics of data** in an enterprise
- The concept of Weak Entity is used to **capture semantics of “header → detail”** records in many real world situations;
  - **‘the header’** captures data **common** across all forms
  - **‘the detail’** captures data **specific** to individual items
    - e.g.,
      - Order → OrderItems
      - Invoice → InvoiceItems
      - Book Title → Book Copy
      - Course → Class
      - DVD Title → DVD Copy

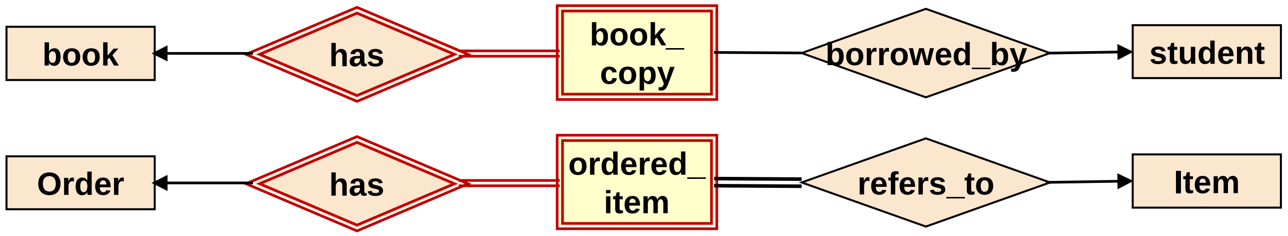
A Guideline of Using Weak Entity Set vs Multivalued Composite Attribute

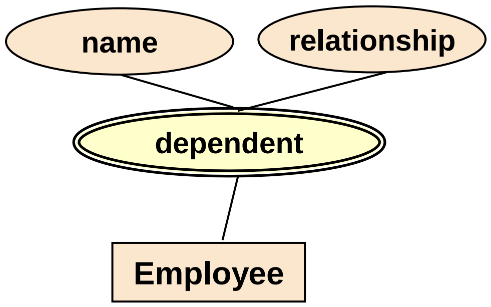
WEAK ENTITY SET

- The entity set **participates in relationships <u>other than</u>** the *identifying relationship*, and
- The entity set has **several** attributes

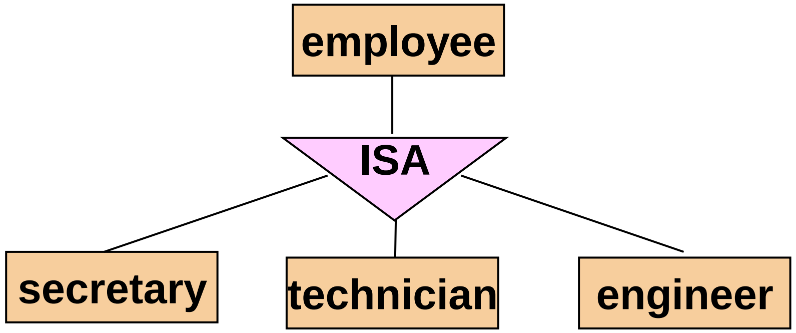
MULTIVALUED COMPOSITE ATTRIBUTE

- The entity set is **participating in <u>only</u>** the *identifying relationship*, and
- The entity set has **few** attributes





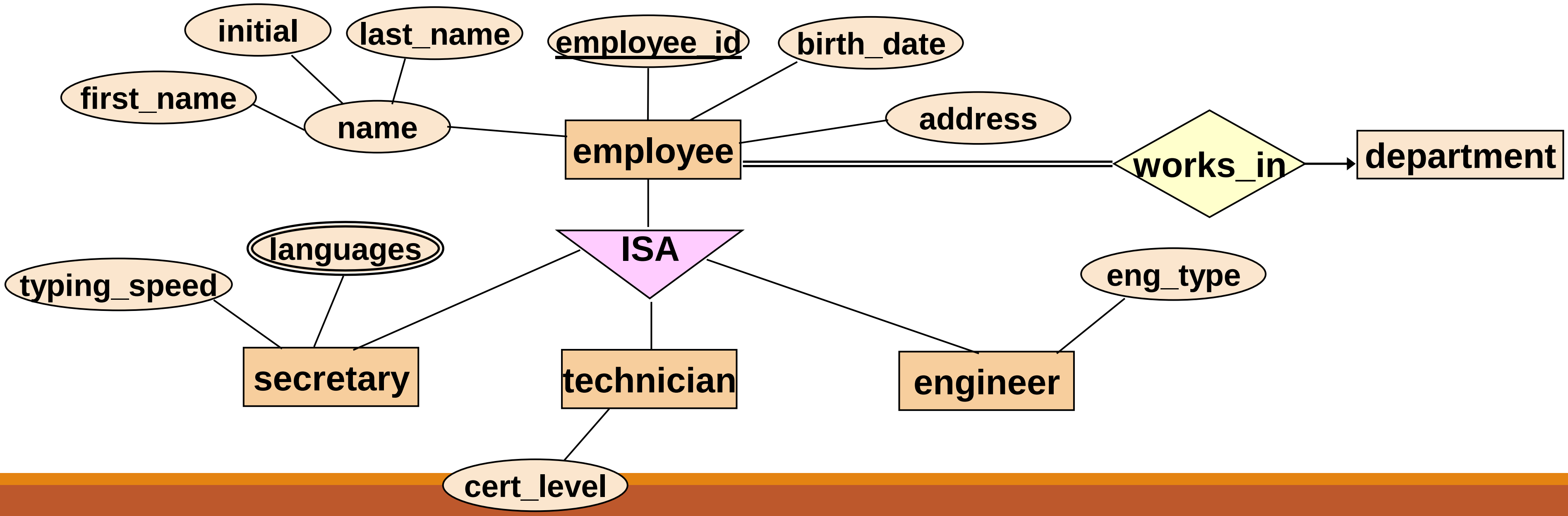
Entity Hierarchy: Superclass and Subclass



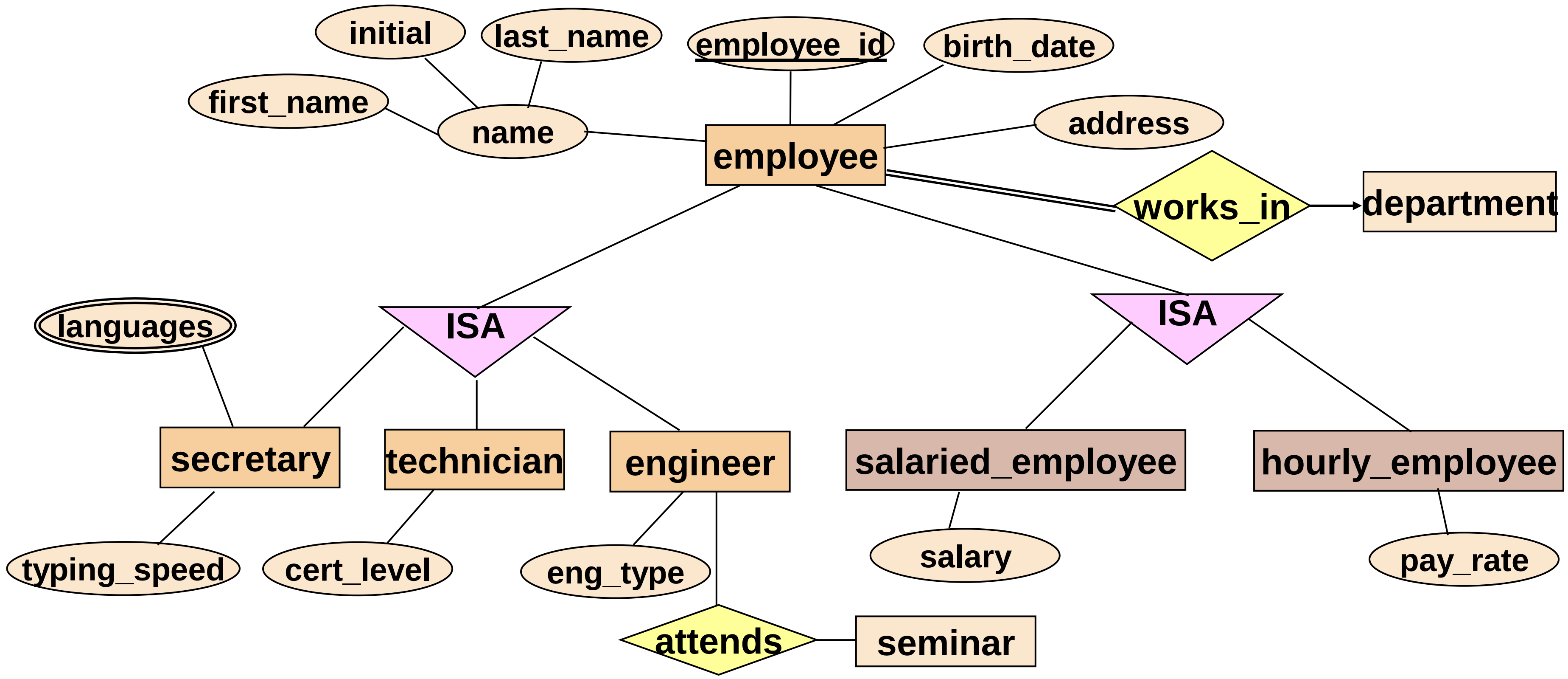
- Allows representation of a set of entities which have **sub-groupings**
- Each of the sub-groupings is called a subclass
  - secretary, engineer, technician *are subclasses of* employee
- The encompassing entity set is called a superclass
  - employee *is a superclass of* secretary, engineer and technician
- Represents **ISA relationship** in OO sense
  - Secretary is an employee; **but** an employee *may not be* a secretary

Attribute Inheritance

- *Subclass* entity set **inherits the following things** from the *superclass* entity set
  1. **all the attributes**
  2. all relationships in which the superclass participates
     - Secretary, technician, engineer entities participates in '*works_in*'

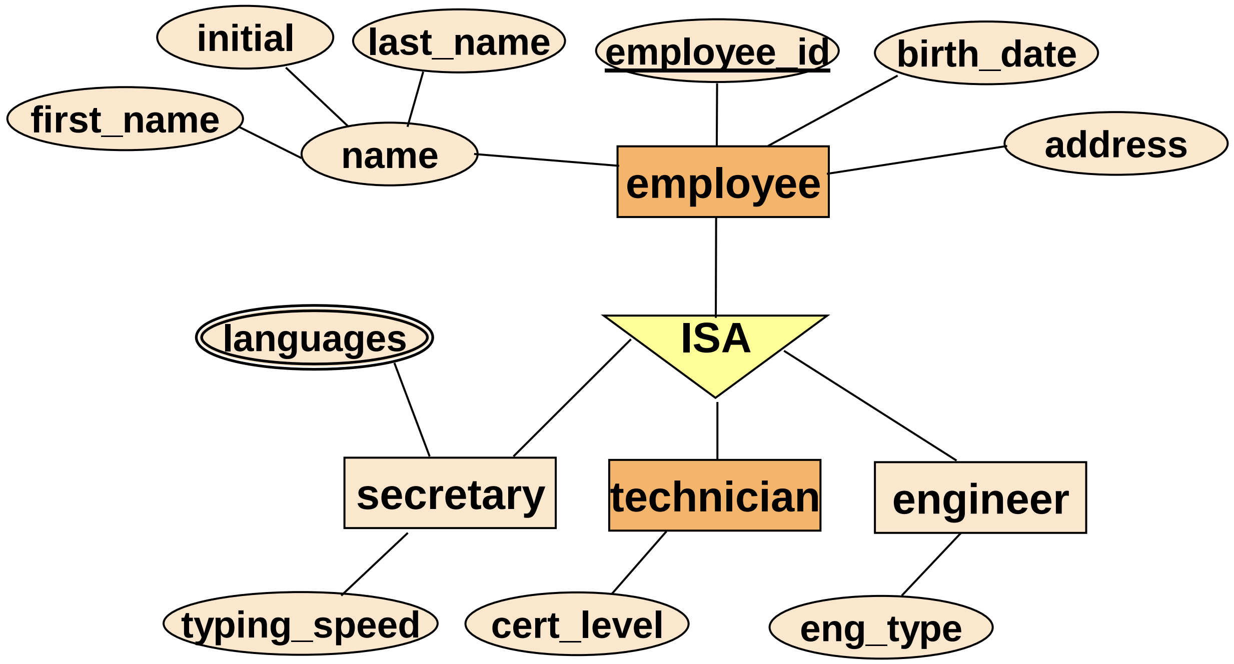


Inheritance Hierarchy Example



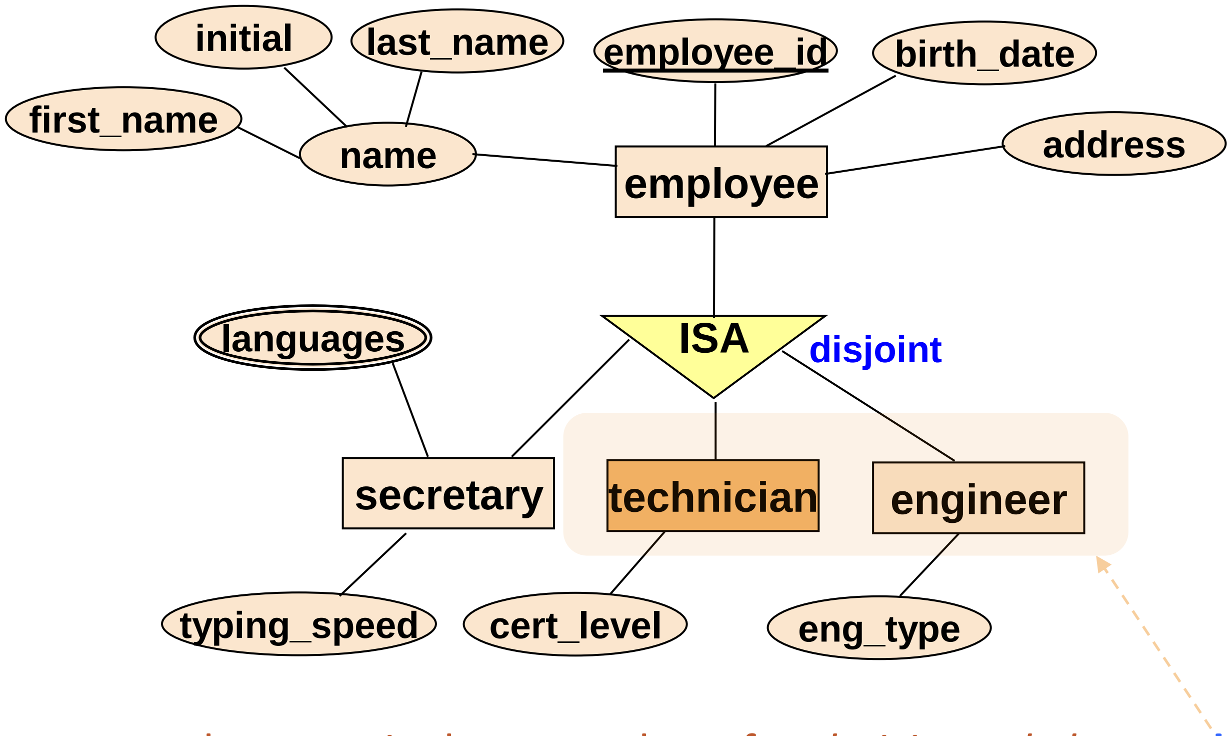
- employee → secretary, technician, engineer
- employee → salaried_employee, hourly_employee
    - e.g) John Doe is a salaried engineer employee

Entity Hierarchy: Superclass and Subclass



- **If** an entity is a *member of a subclass*, it **must also be** a *member of the superclass*.
- An entity **may** *belong to* ***only one / several*** *sub-classes* (**Disjoint** vs. **Overlapping**)
- An entity **need / need not** *belong to a subclass* (**Total** vs. **Partial**)

Disjointness Constraints



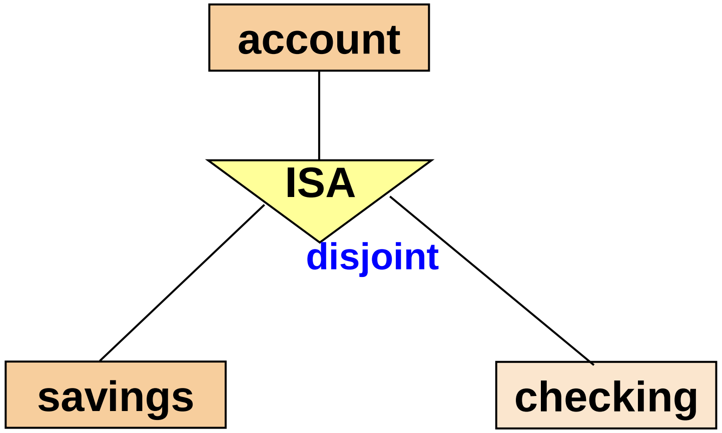
- Can an employee entity be a member of *technician subclass* **and also be** a member of *engineer subclass*?
  - Disjoint: an entity *belong to* **no more than one lower-level** *entity set*
  - Overlapping (default): an entity *may belong to* **more than one lower-level** *entity set*

Another Example: Disjoint vs Overlapping

DISJOINT

- An *account* entity **must be either** a member of *savings* entity set, **or** a member of *checking* entity set

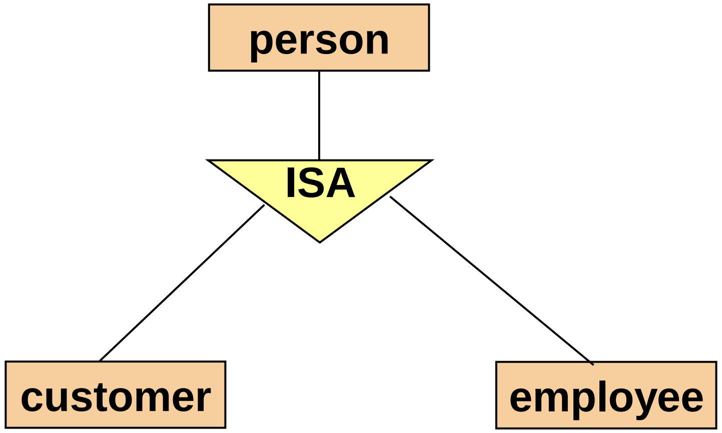
(at most ‘**one**’, not both)



OVERLAPPING

- A person entity can be a member of **one or both** *customer* and *employee* entity sets

(at most ‘**all**’)



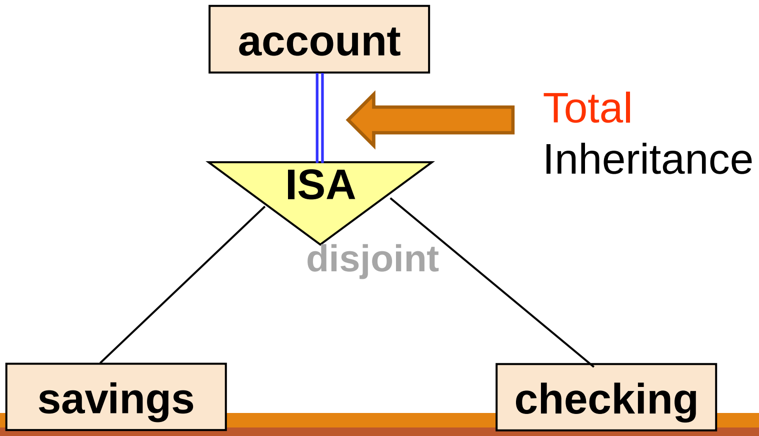
Completeness Constraints

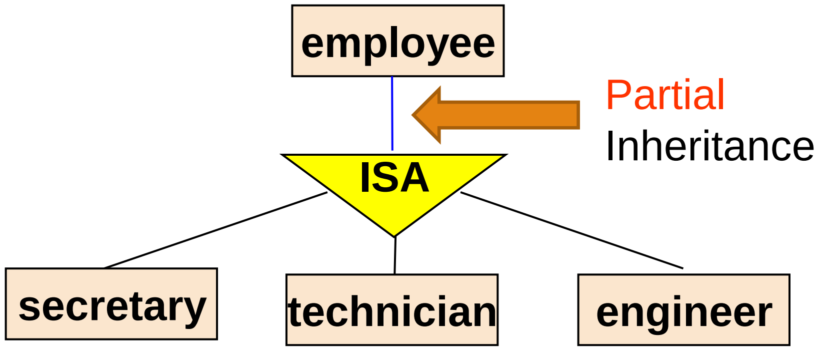
TOTAL INHERITANCE

- Each higher-level entity **must belong to** at least one *of the lower-level entity set* *(at least 1)*
  - Denoted by **double line** from the superclass to the triangle in ER diagram

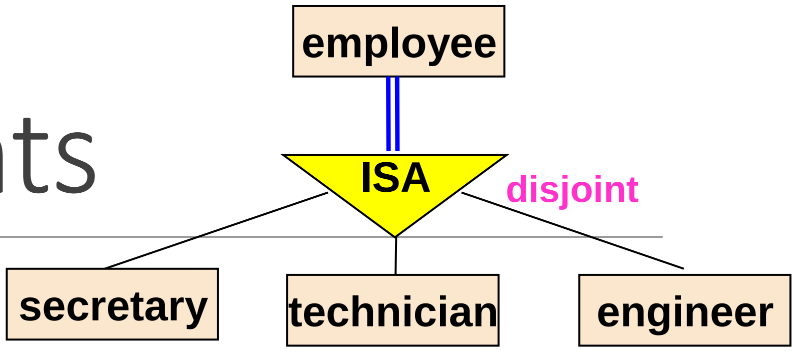
PARTIAL INHERITANCE

- Some higher-level entities **may not belong to** any *lower-level entity set* *(at least 0)*
  - Default
  - Denoted by **single line**



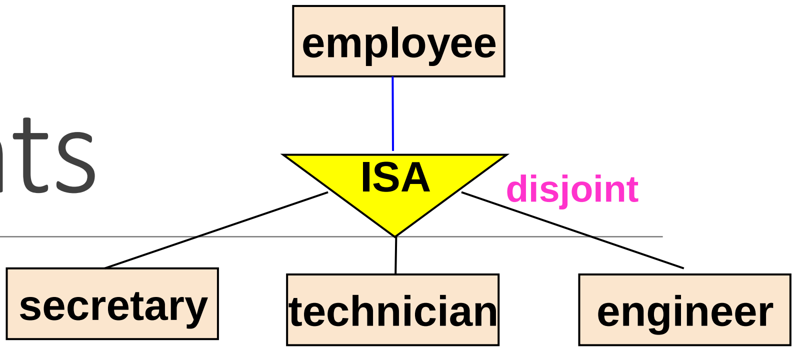


Inheritance Variants



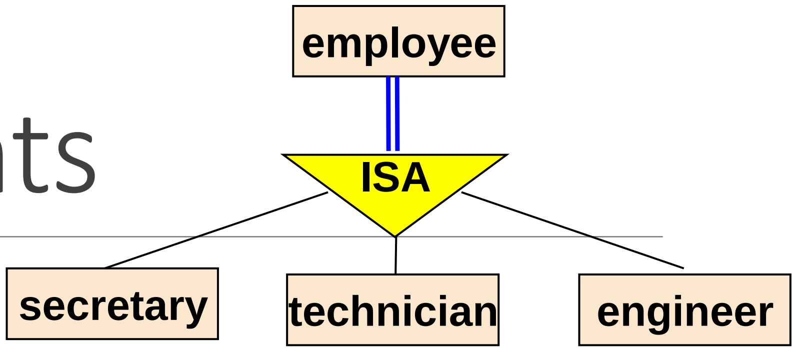
- Total-Disjoint
  - Every entity in the superclass **must belong to only one** of the subclasses
- Partial-Disjoint
  - An entity in the superclass **may or may not belong to** a subclass
    - If an entity belongs to a subclass, it must **belong to** only one subclass
- Total-Overlapping
  - Every entity in the superclass **must belong to one or more** subclasses
- Partial-Overlapping
  - An entity in the superclass **may** or may not **belong to one or more** subclasses

Inheritance Variants



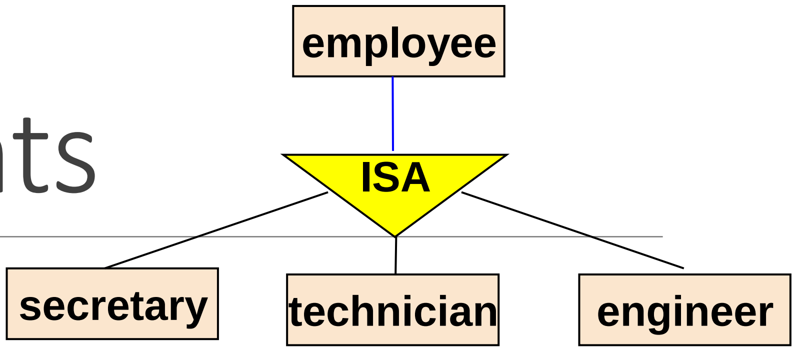
- Total-Disjoint
  - Every entity in the superclass **must belong to only one** of the subclasses
- Partial-Disjoint
  - An entity in the superclass **may or may not belong to** a subclass
    - If an entity belongs to a subclass, it must **belong to** only one subclass
- Total-Overlapping
  - Every entity in the superclass **must belong to one or more** subclasses
- Partial-Overlapping
  - An entity in the superclass **may or may not belong to one or more** subclasses

Inheritance Variants



- Total-Disjoint
  - Every entity in the superclass **must belong to only one** of the subclasses
- Partial-Disjoint
  - An entity in the superclass **may or may not belong to** a subclass
    - If an entity belongs to a subclass, it must **belong to** only one subclass
- Total-Overlapping
  - Every entity in the superclass **must belong to one or more** subclasses
- Partial-Overlapping
  - An entity in the superclass **may or may not belong to one or more** subclasses

Inheritance Variants



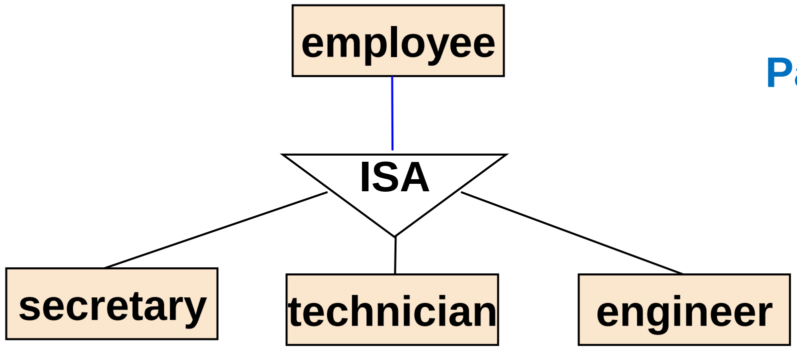
- Total-Disjoint
  - Every entity in the superclass **must belong to only one** of the subclasses
- Partial-Disjoint
  - An entity in the superclass **may or may not belong to** a subclass
    - If an entity belongs to a subclass, it must **belong to** only one subclass
- Total-Overlapping
  - Every entity in the superclass **must belong to one or more** subclasses
- Partial-Overlapping
  - An entity in the superclass **may or may not belong to one or more** subclasses
    - There may be entities in the superclass which **may not belong to any** of the subclasses

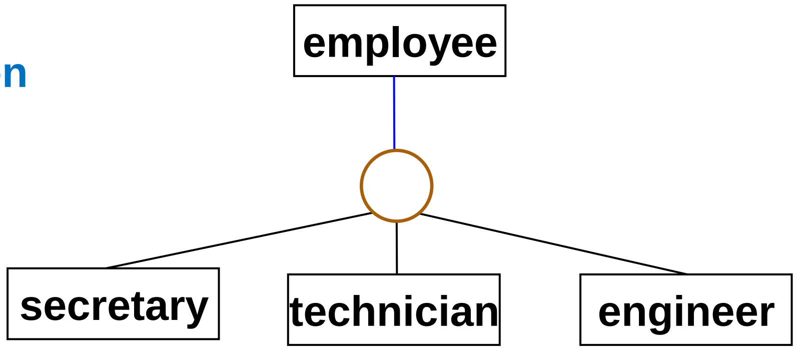
Alternative Notation - 1

Classical Notation

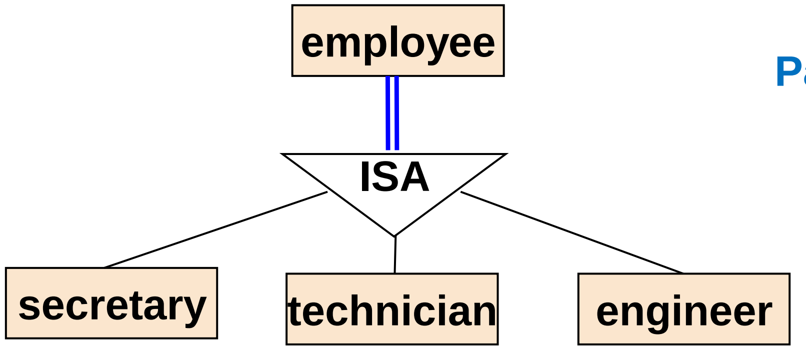
Alternative Notation

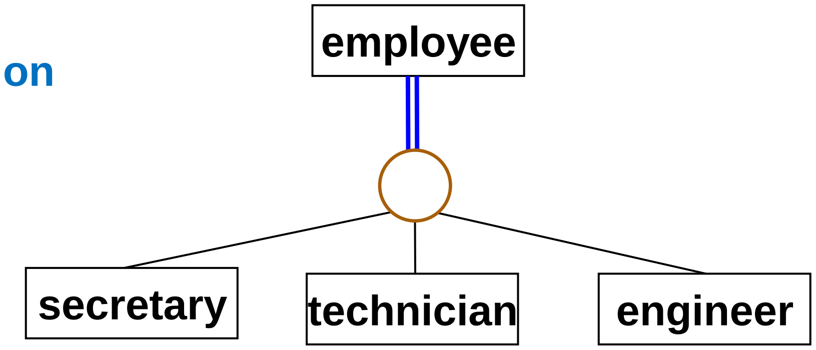
Partial Participation





Total Participation



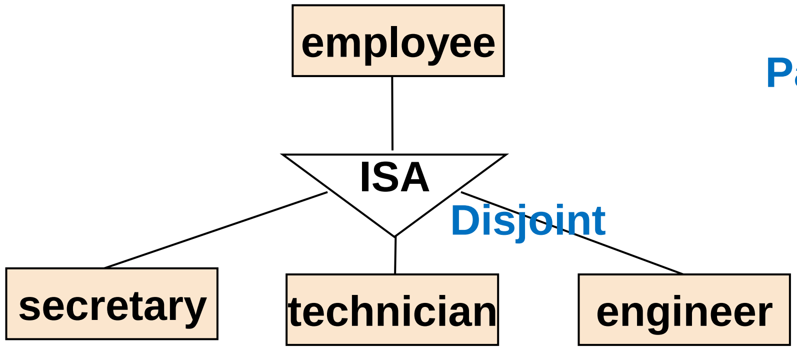


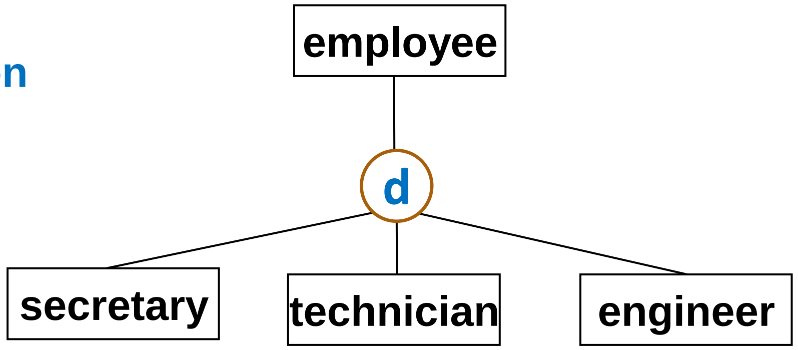
Alternative Notation - 2

Classical Notation

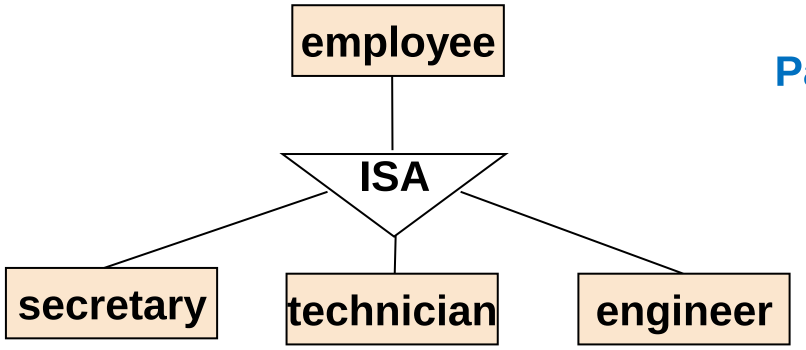
Alternative Notation

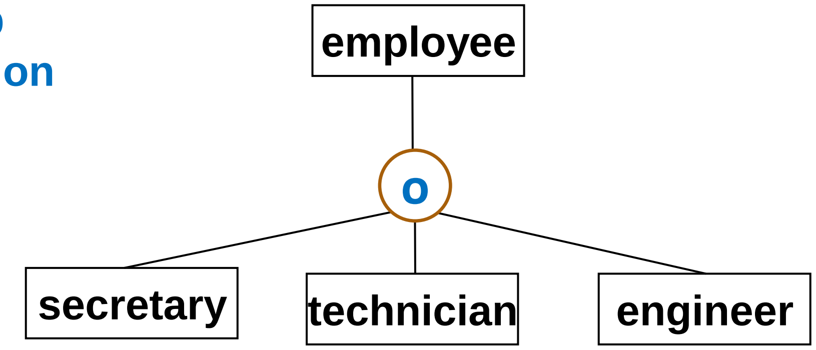
Disjoint Participation





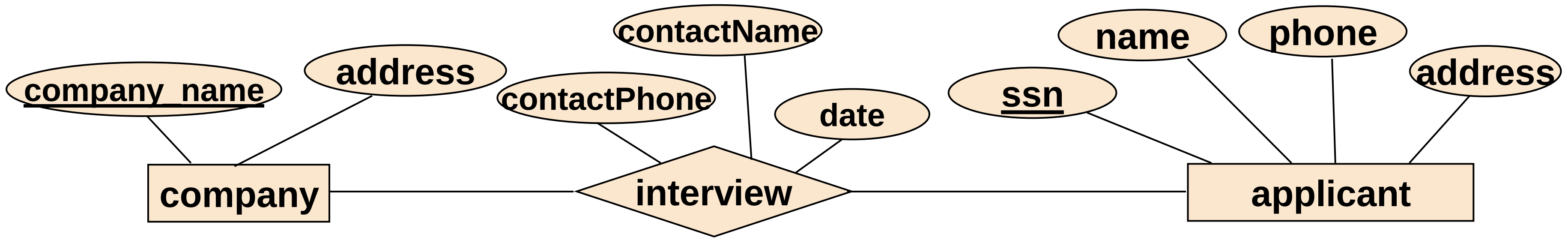
Overlap Participation



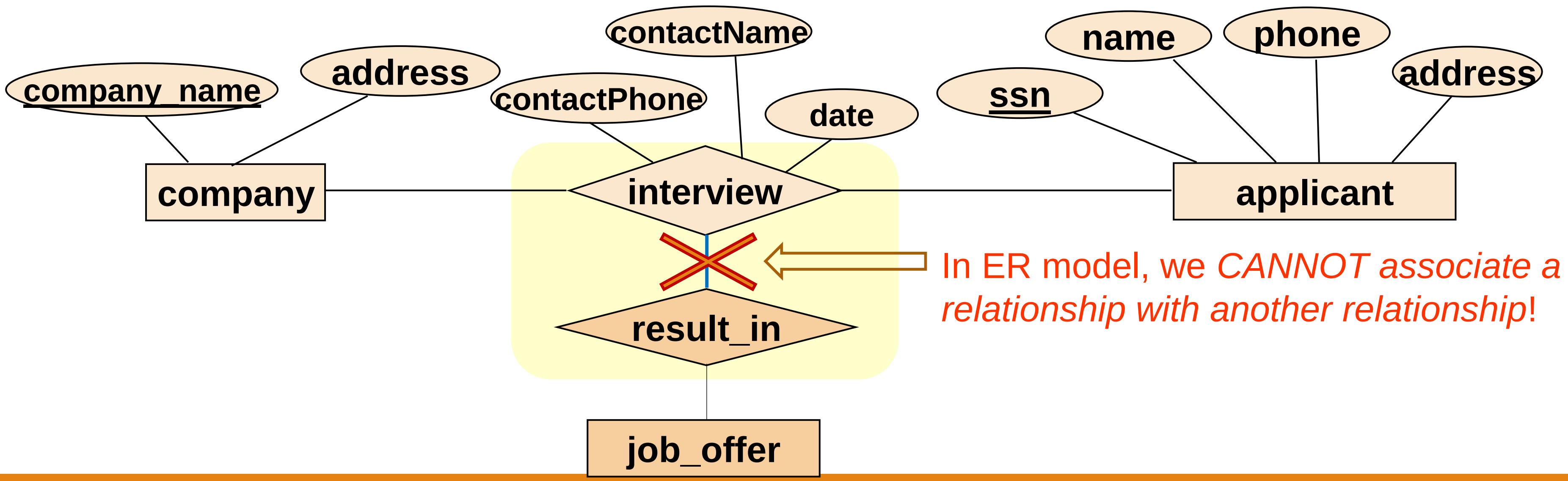


Aggregation (Entity Cluster) - 1

- Relationship: association among entities
- Relationship with other relationships? Cannot be expressed in ER model

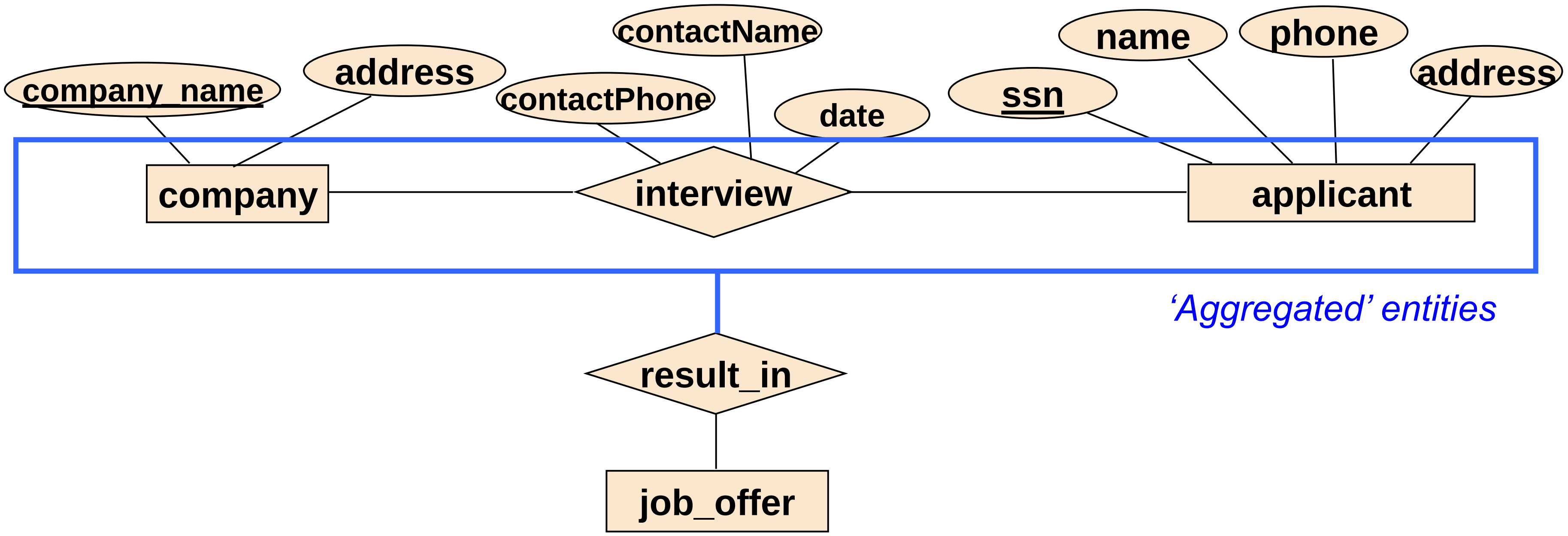


- Suppose some interviews result in job offers, while others do not



In ER model, we *CANNOT associate a relationship with another relationship*!

Aggregation (Entity Cluster) - 2

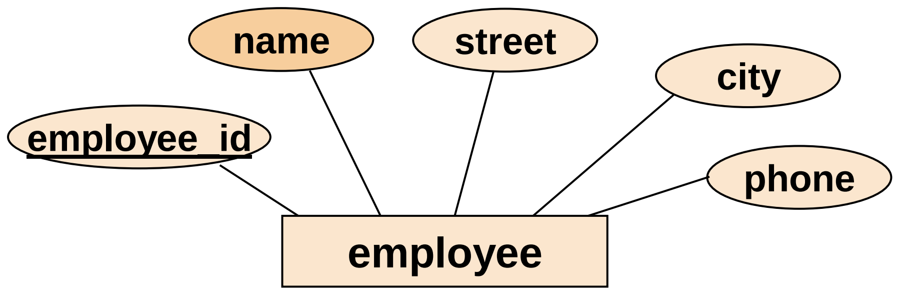


- **Aggregation** allows *relationships to be abstracted* as entities *(represented by using a **rectangle** that contains a relationship and associated entities)*
  - Such 'aggregated' entities can be associated with other entities through another relationship set
    - e.g.) Allows to reflect the fact that some interviews may result in job offers while others do not

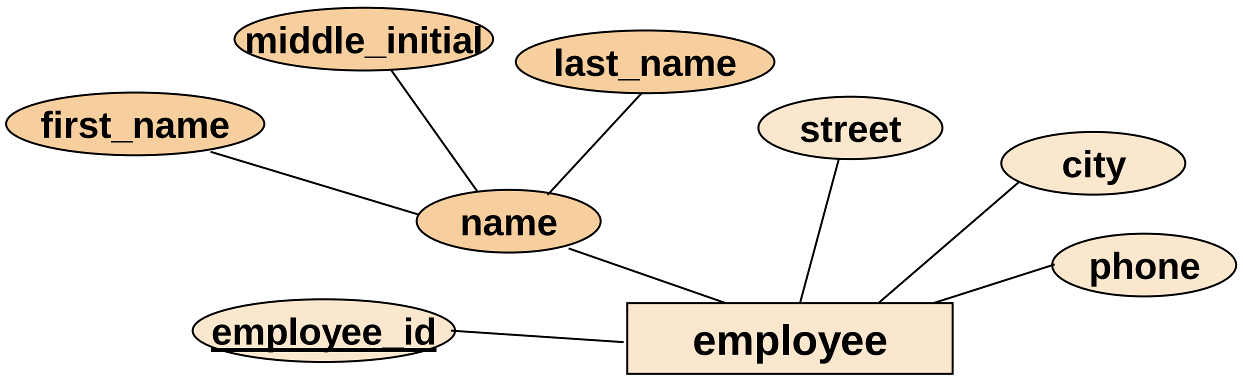
Design Alternatives

- Simple Attribute vs. Composite Attribute
- Entity Sets vs. Attributes
- Entity Sets vs. Relationship Sets
- Attributes vs. Relationship Sets
- Placement of Descriptive Attributes
- Weak Entity Sets vs. Strong Entity Sets
- Weak Entity Sets vs. Multi-valued Attributes
- Generalization/Specialization Hierarchy vs. Separate Entities
- N-ary relationships vs. multiple binary relationships

Simple Attribute vs. Composite Attribute

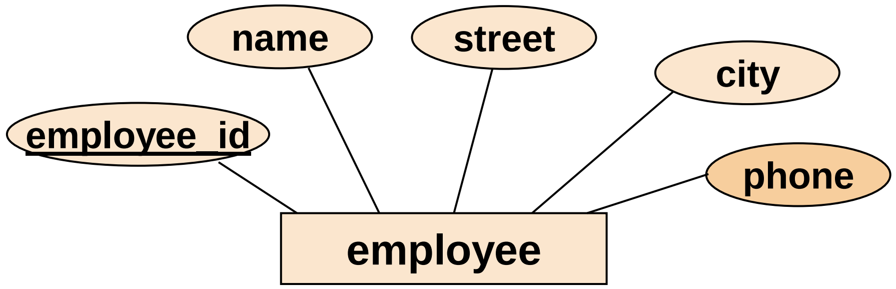


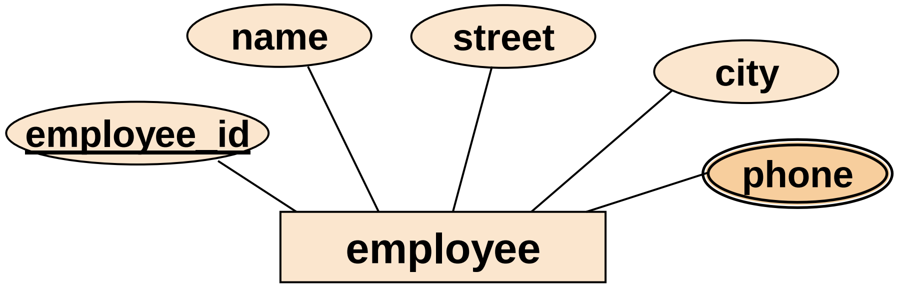
- What's the difference?
- When to represent something as a simple attribute?
- When to represent something as a composite attribute?

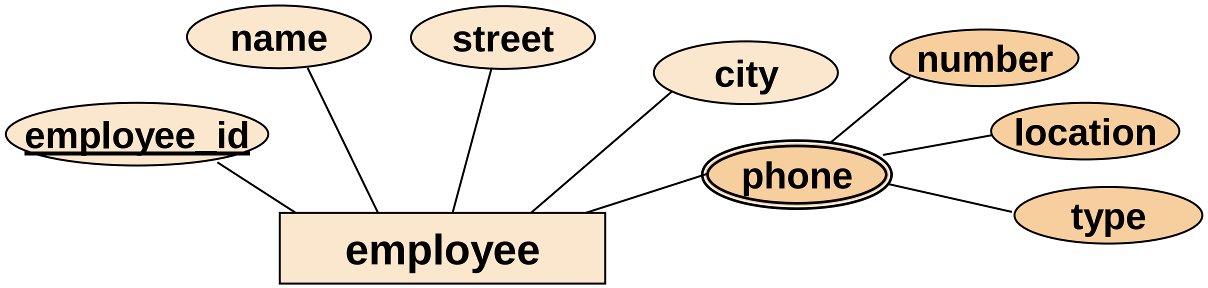


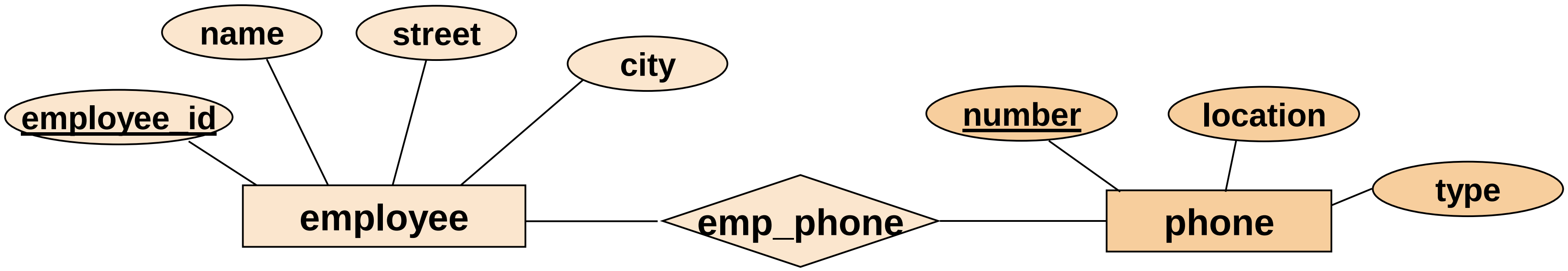
Entity Sets vs. Attributes

- Is phone an attribute or entity set?
  - What are the differences?

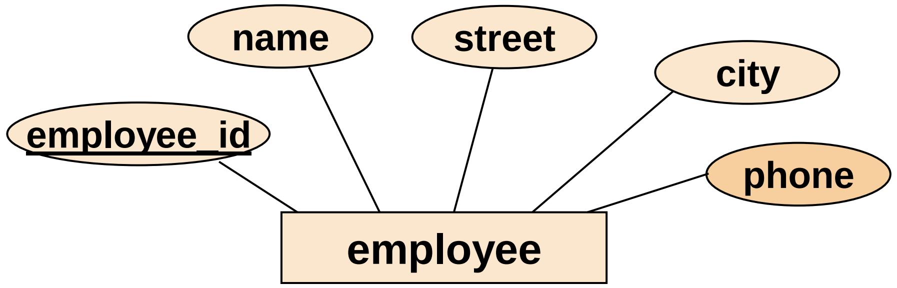






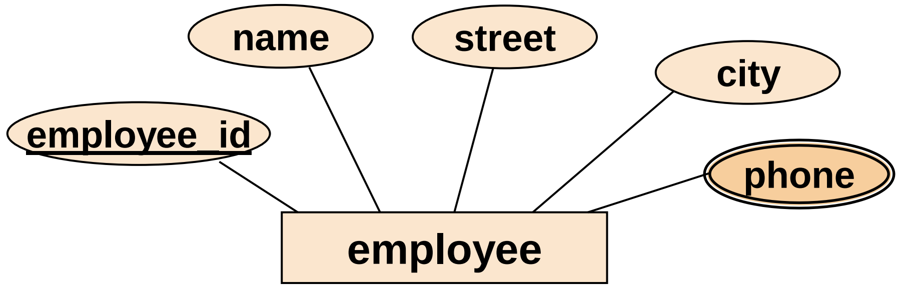


Phone as a simple attribute



- CAN:
  - Employee must have a phone number
    - Null can be used to say no phone number; but null complicates things
- CANNOT:
  - Employee cannot have more than one number
  - No other information about phone can be maintained

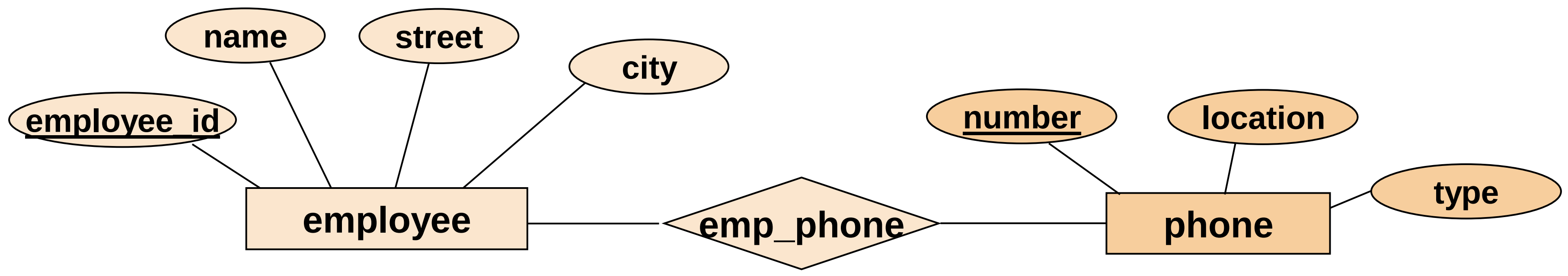
Phone as a multi-valued attribute



- CAN:
  - Employee may have more than one phone number

- CANNOT:
  - No other information about each phone number can be maintained

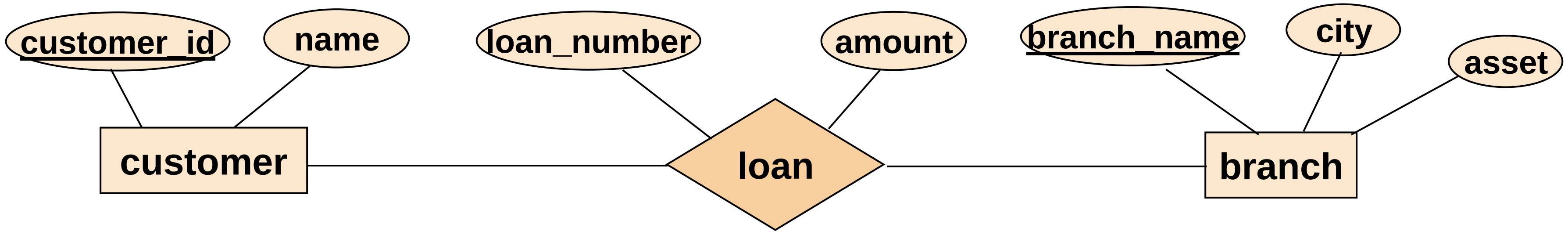
Phone as a separate entity set



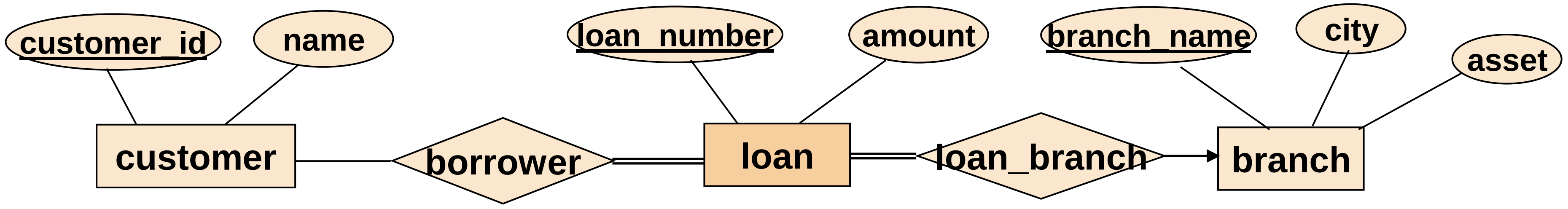
- CAN:
  - Employee can have zero, one, or more phone numbers
    - Minimum, Maximum Cardinality on the relationship set can provide more control and flexibility
  - Can maintain extra information about each phone number
    - Location, type (mobile, video phone, VoIP, etc)
  - The separate 'Phone' entity can also participate in other relationships!
  - More flexibility and more extensibility

Entity Sets vs. Relationship Sets

loan as a relationship set

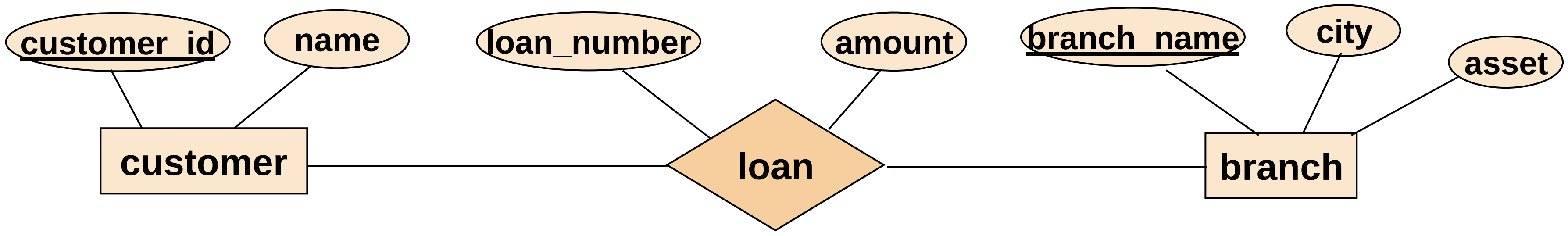


loan as an entity set



loan as a relationship set

- Limitation: a **customer** can hold **only a single loan** on **a branch**
  - A customer cannot hold multiple loans on the same branch.
  - Joint loan holders are not allowed.

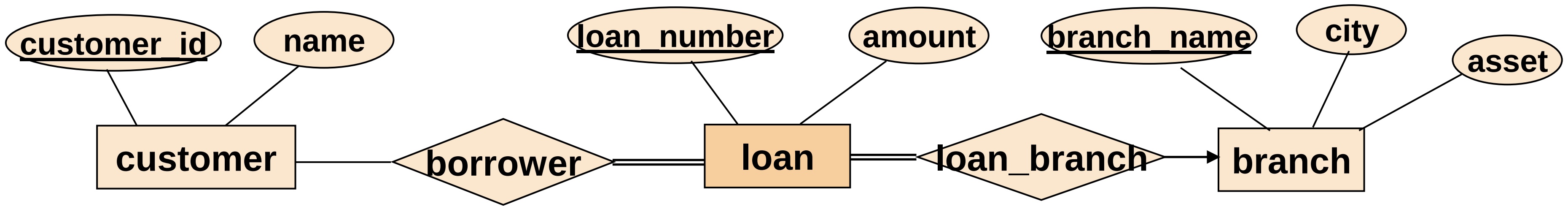


e.g.,
(**c100**, L-17, 1000, **Downtown**)
(**c100**, L-18, 1500, **Downtown**)??

Note: descriptive attributes are **not** used to **differentiate** relationship's instances.
Cons: store duplicate values of loan's amount → Data Inconsistency

loan as an entity set

- Enhancement:
  - a customer can have **multiple loans in a branch** or in multiple branches
  - *Joint loan holders are also allowed.*
    - Business logic can be changed easily to enforce "No jointly held loans" by *simply changing the cardinality mapping*



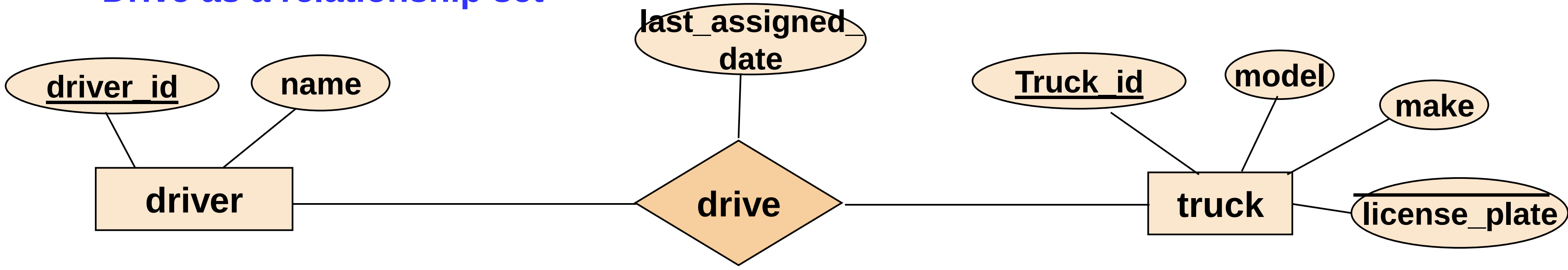
e.g.,
(c100, L-17)
(c100, L-18)

e.g.,
(L-17, Downtown)
(L-18, Downtown)

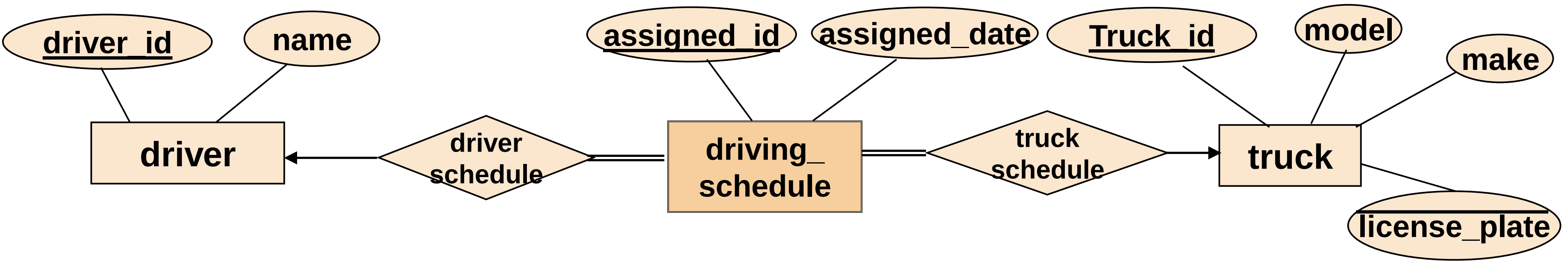
Enhancement: store **one** amount's value **per loan**

Another Example:
Driver and Truck

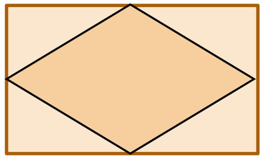
Drive as a relationship set



Driving_schedule (drive) as an entity set



Associative Entity

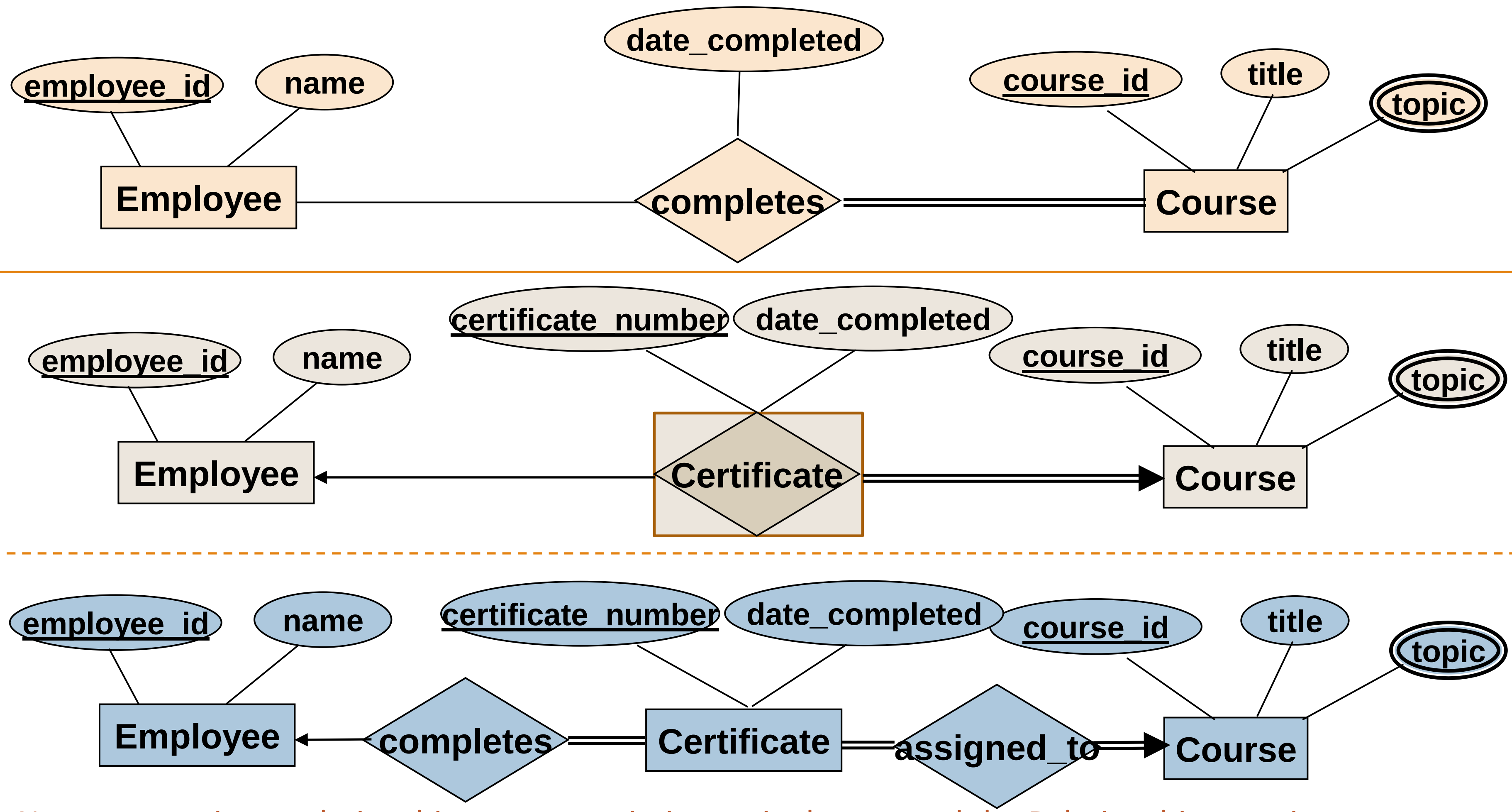


- An entity type that associates the instances of one or more entity types *and contains attributes* ***that are peculiar*** *(particular)* ***to the relationship*** between those entity instances.

- **Guidelines** to check whether to **convert** a relationship **to an associative entity type**?

1. **All** the relationships for the participating entity types are **"many" relationships**.
2. The resulting associative entity set has **independent meaning to end users** and, preferably, can **be identified with a single-attribute identifier**.
3. The associative entity has **one or more attributes in addition** to the identifier.
4. The associative entity **participates in one or more relationships** ***independent of*** *the entities related in the associated relationship*.

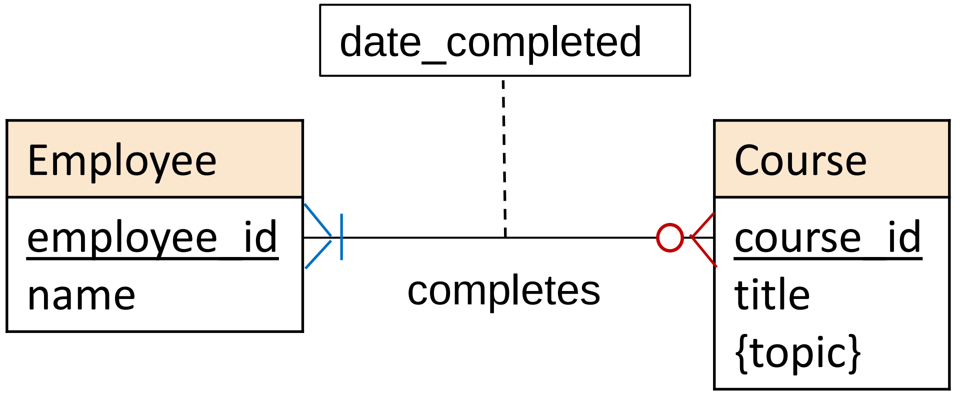
Associative Entity is used to resolve many-to-many relationships

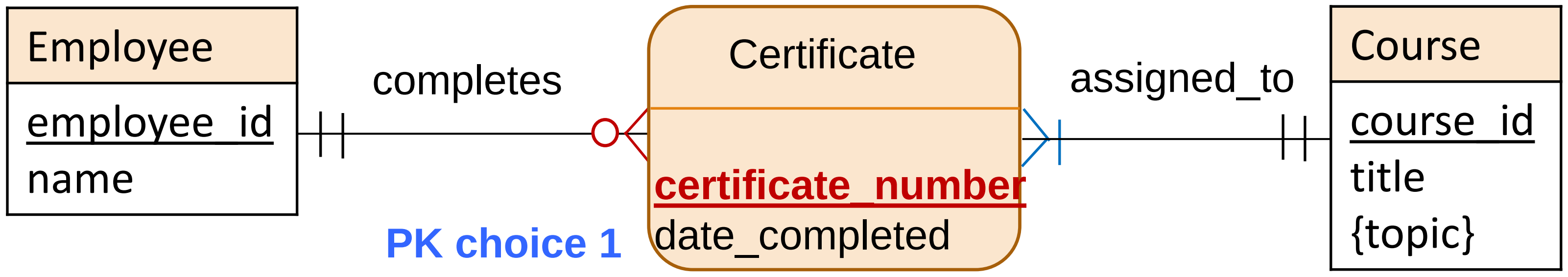


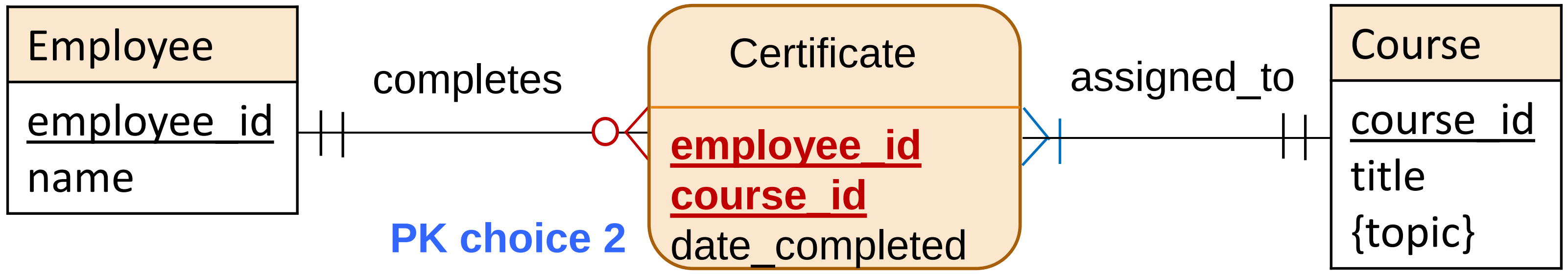
Note: converting a relationship to an associative entity has caused the Relationship notation to move.

- the "many" cardinality now terminates at the associative entity, rather than at each participating entity type.

Associative Entities: Alternative Notation

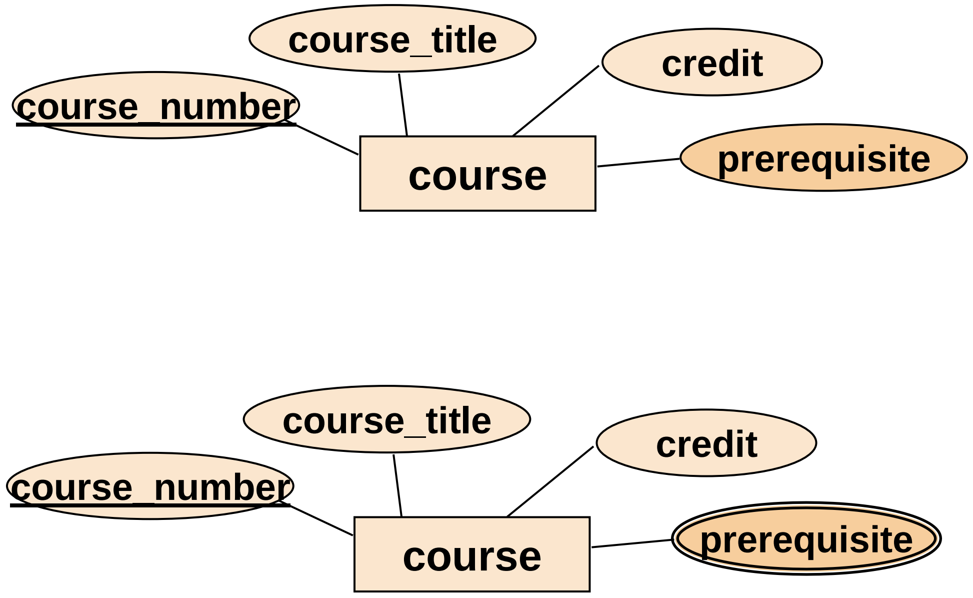


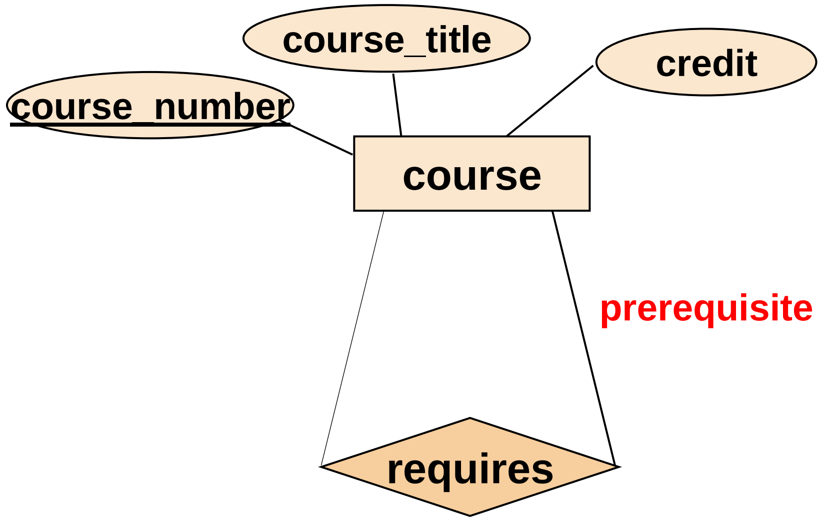




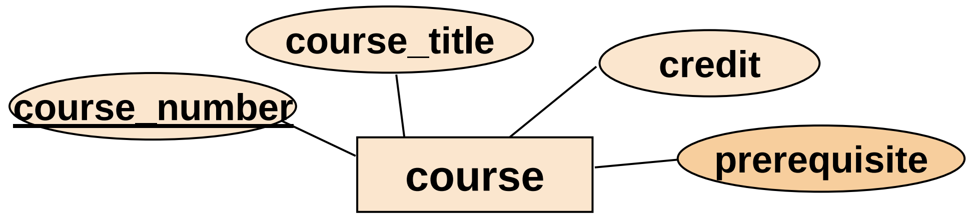
Attributes vs. Relationship Set - 1

- Is Prerequisite an attribute or relationship set?
  - What are the differences?



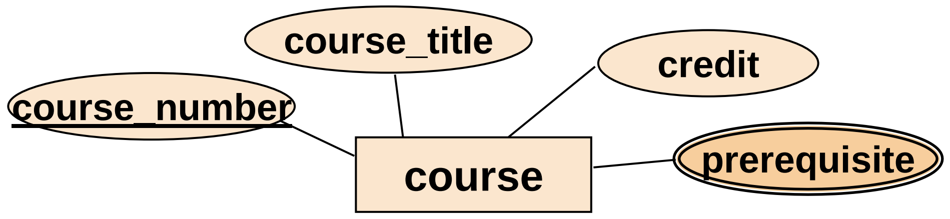


Prerequisite as an attribute



- CAN:
  - A course **has only one prerequisite**
    - need to use null for a course *without* a prerequisite
- CANNOT:
  - Cannot enforce the **valid** prerequisite courses

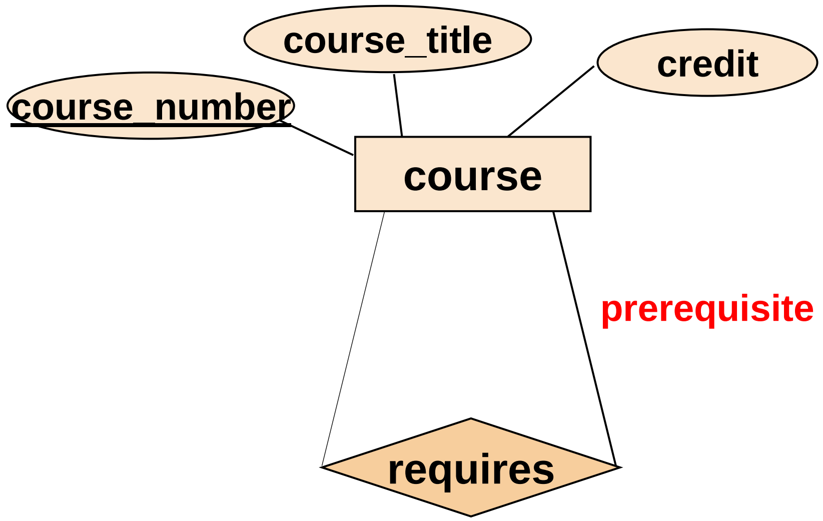
Prerequisite as a multi-valued attribute



- CAN:
  - Can support **several** prerequisites
    - Still need to use null for a course without a prerequisite

- CANNOT:
  - Cannot make sure of the **valid** prerequisite courses

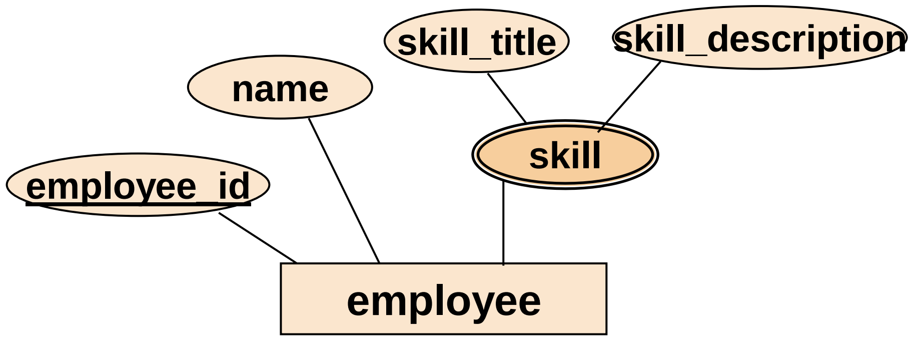
Prerequisite as a relationship set

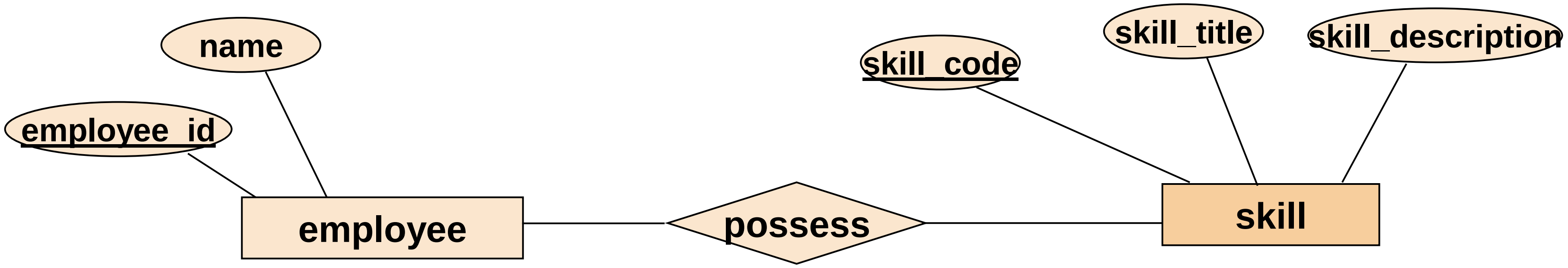


- CAN:
  - Support **several** prerequisites
    - No need to use null for a course *without* a prerequisite
  - **Make sure a prerequisite is a valid existing course**

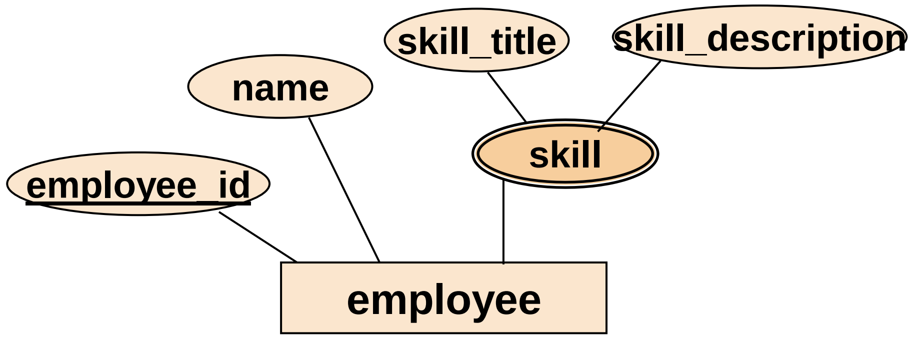
Multivalued Attributes vs. Entity Set

- Is Skill a multi-valued attribute or entity set?
    - What are the differences?



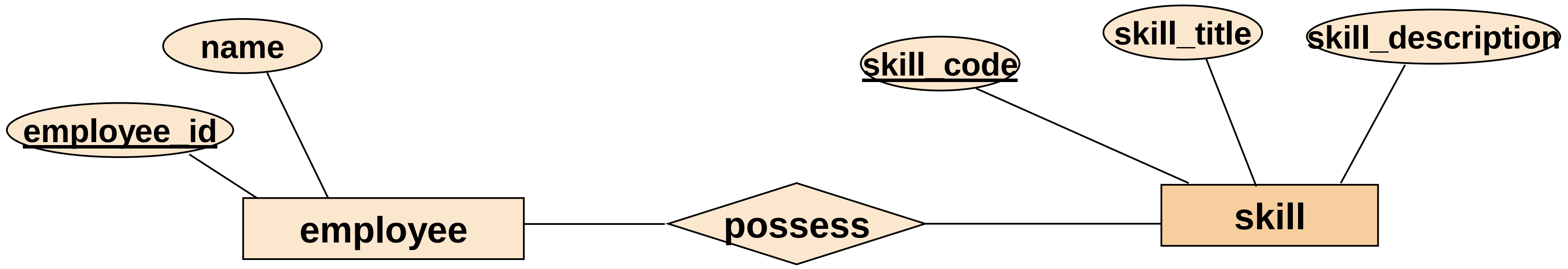


Skill as a multi-valued attribute



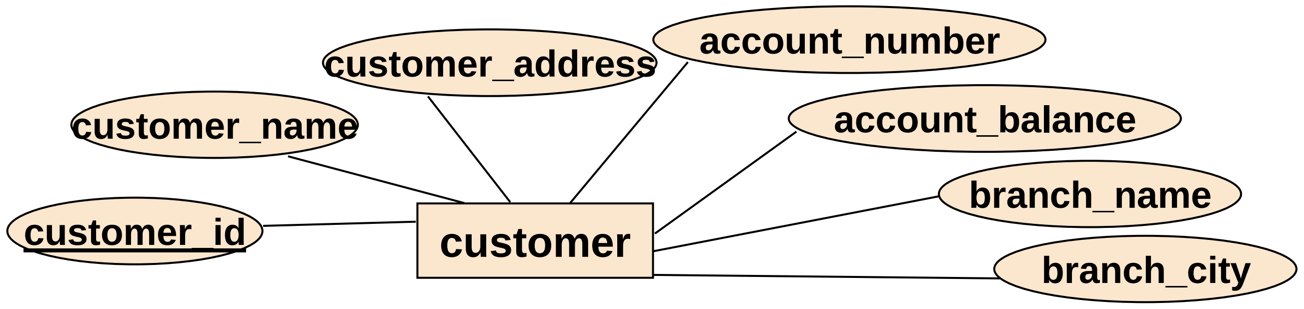
- CAN:
  - One employee may have more than one skill.
- CANNOT:
  - Skill cannot be associated with any other entities
    - Possible drawback:
      - Skill information can suffer from **Data Redundancy** (therefore possible **Data Inconsistency**)
      - Skill are just attributes; therefore, we can have **different values** even **for** logically **same** skill.

Skill as an entity set



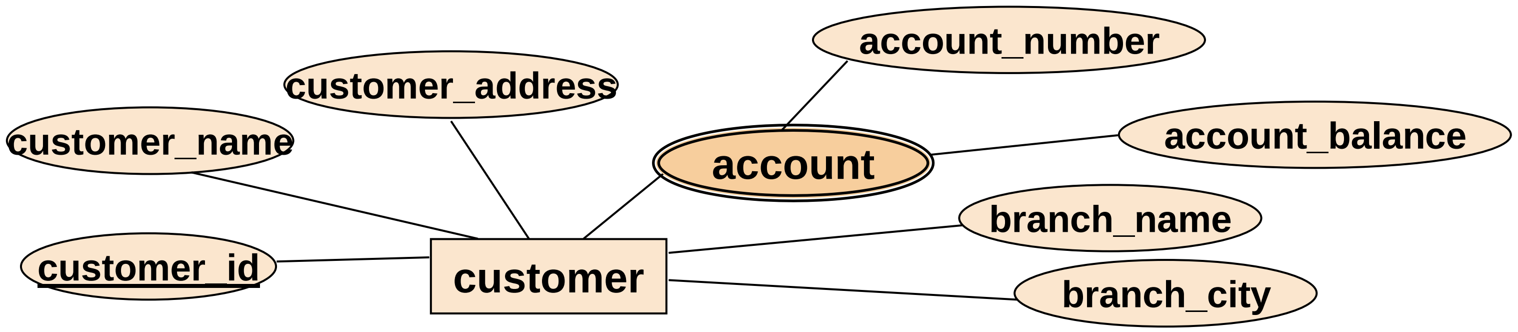
- ENHANCEMENT:
  - Skill can be *associated with any other entities*
    - e.g., skill entities can be associated with project entities
    - Offers room for flexibility and growth

Small Design Exercise - 1



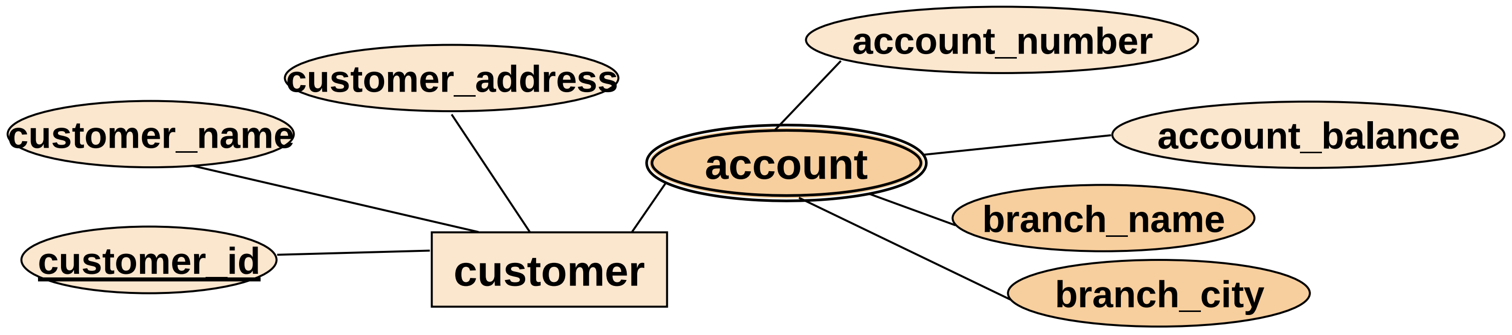
- A customer is always required to have **only one account in one branch**
  - Can a customer not have any account?
  - Can a customer have more than one accounts?

Small Design Exercise - 2



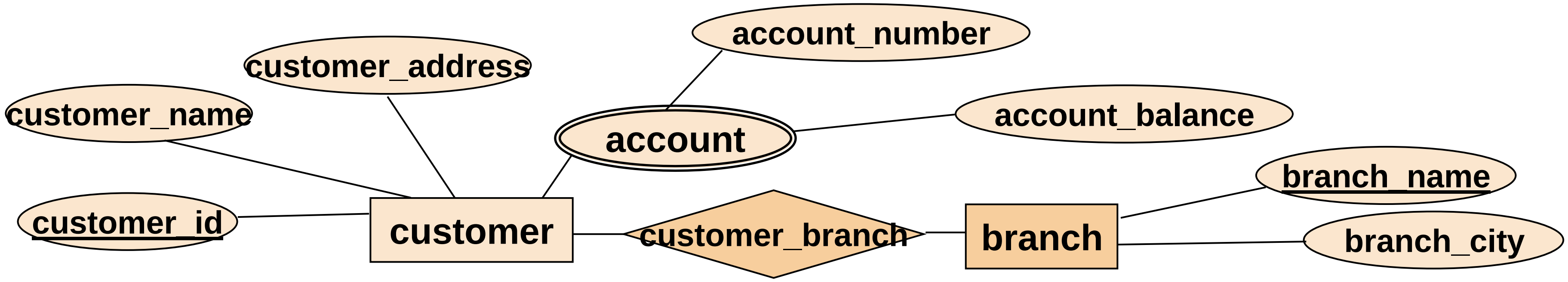
- A customer can have accounts at only one branch
  - Can a customer not have any account?
  - Can a customer have multiple accounts in different branches?
  - What if there are joint account holders?

Small Design Exercise - 3



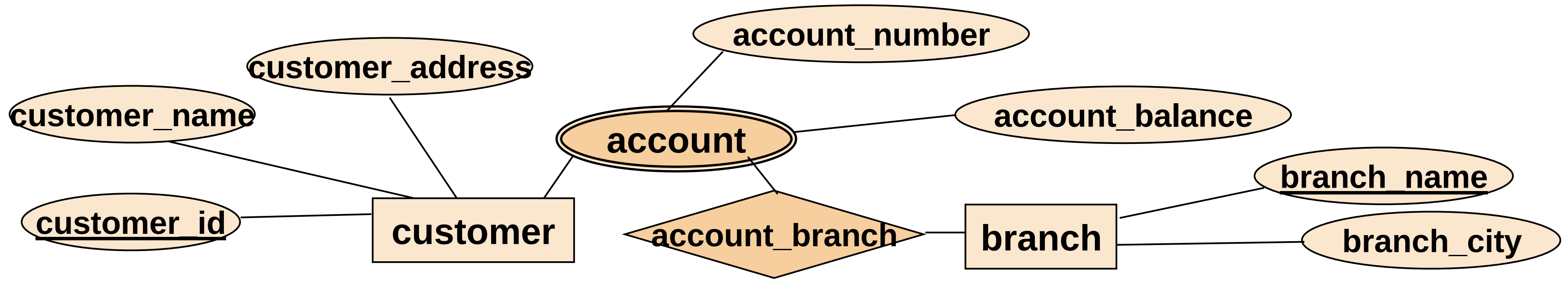
- **Information** about **branch** is *repeated for every account* in the branch!
  - Data Redundancy → Modification Anomaly → Data Inconsistency

Small Design Exercise - 4



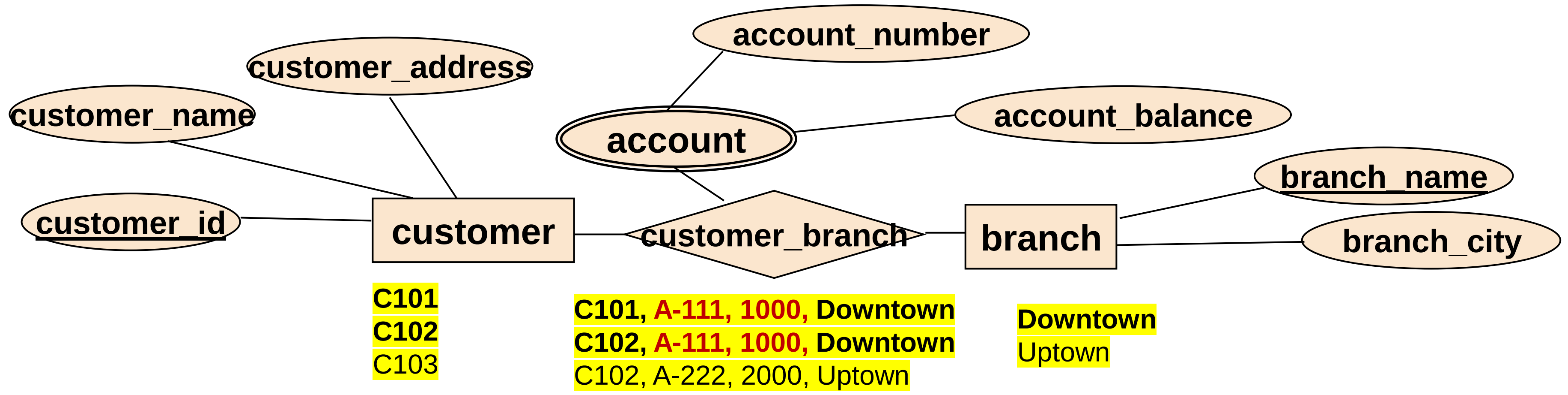
- customer_branch associates a customer entity with a branch entity
  - A customer can be a member of zero, one or more branches
  - BUT, how do you know the branch where the account is opened?

Small Design Exercise - 5



- Is this even a valid ER diagram?

Small Design Exercise - 6

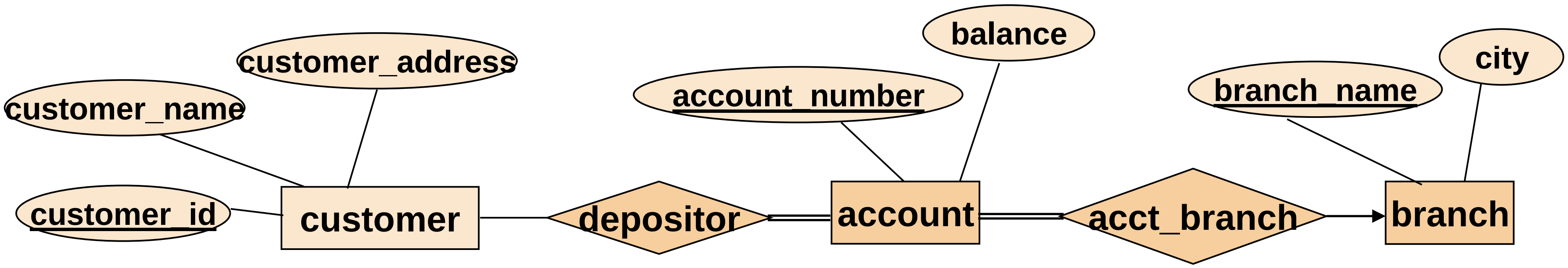


- Can record all the accounts a customer has in each different branch

  (for each customer, at most one account per branch)

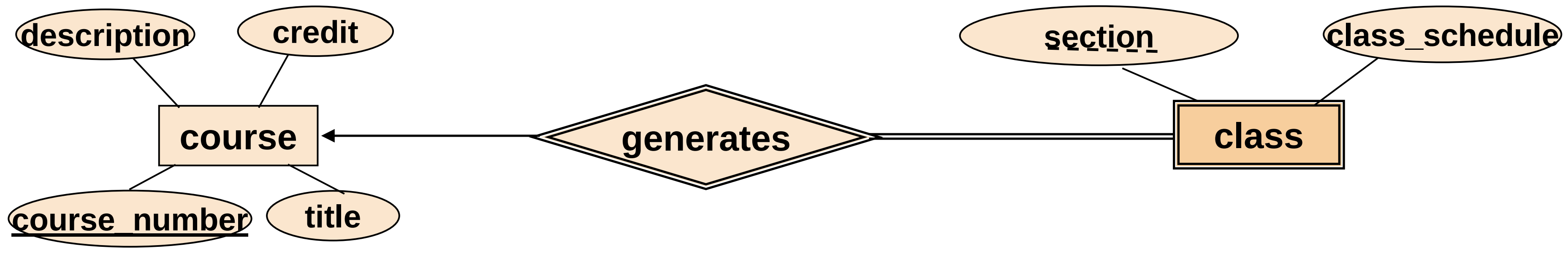
- Can we have joint account holders? Yes, BUT
  - Requires account information (number, balance) to be replicated for each joint holders → Data Redundancy → Possible Data Inconsistency
- Can we create a relationship between an account and other entities (e.g. Auditor)?

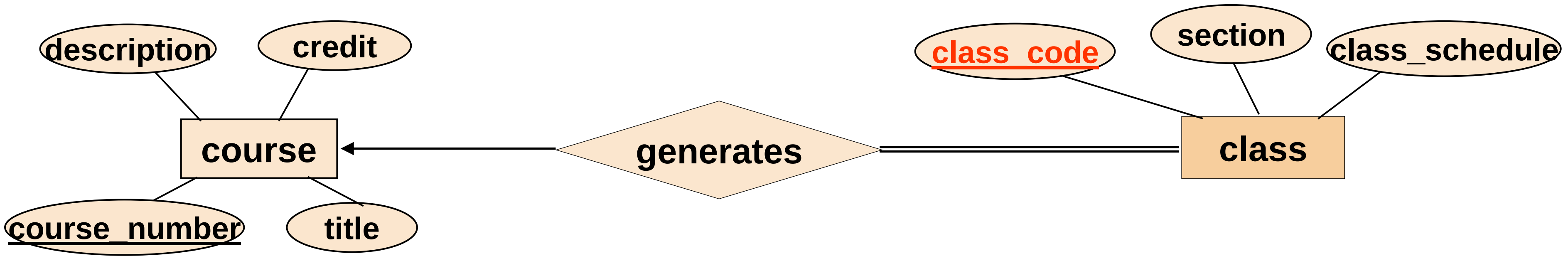
Small Design Exercise



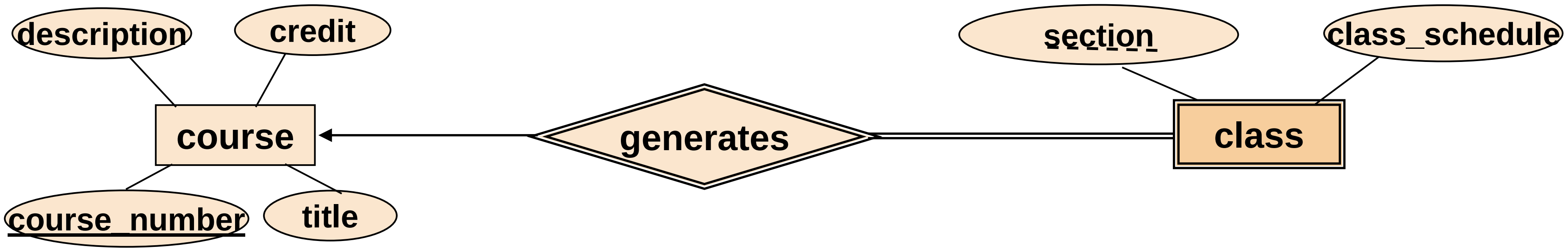
- Allows joint account holders
    - Does not repeat the account information; only one balance maintained
- Customer may not have any account (partial participation)
- Allows customers to have multiple accounts in the same branch
- Allows customers to have multiple accounts in different branches
- Can record which accounts are opened in which branch
- Can record which customers are having what accounts in which branches

Weak Entity Set vs. Strong Entity Set - 1



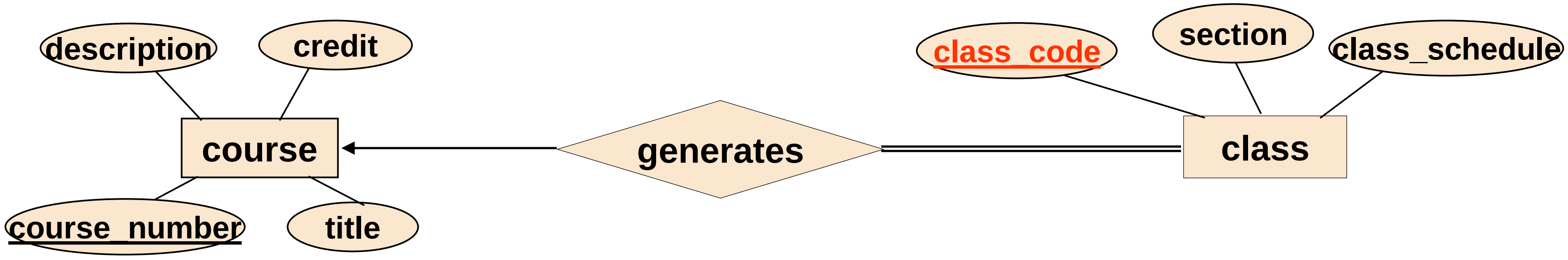


Class as a weak entity set



- Course is the Identifying Entity Set of class
- Existence of class entities **depends on** a course entity

Class as a strong entity set



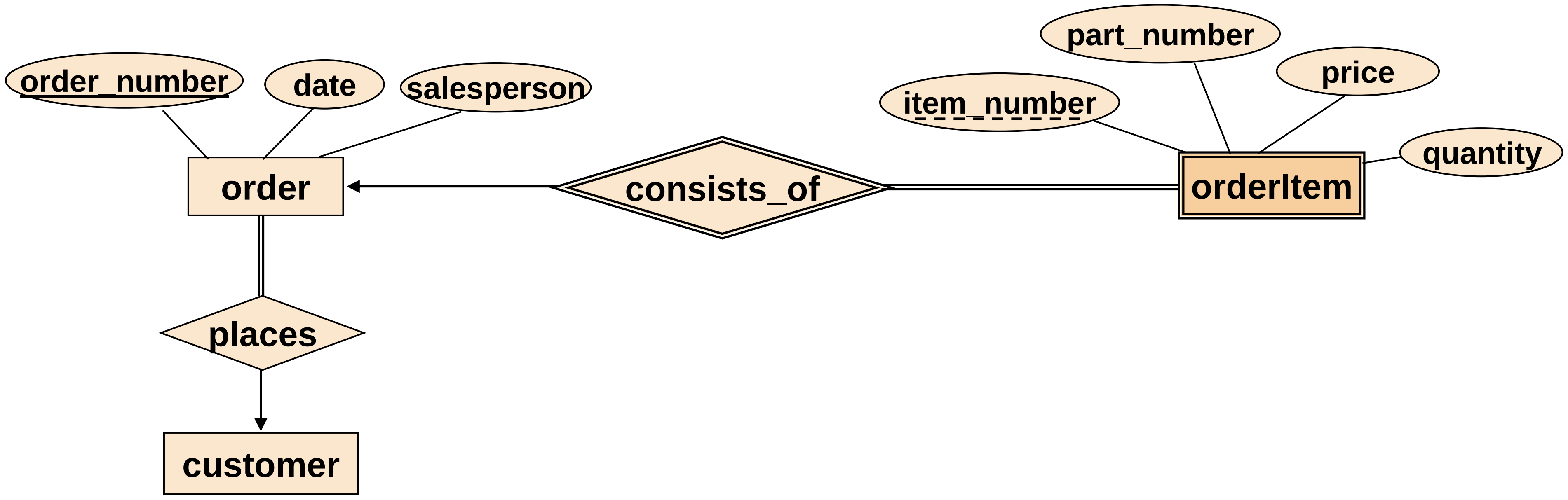
- *class* entity set now has a primary key, *class_code*, which can uniquely identify a class without requiring a *course_number*

- *generates* is NOT an identifying relationship set;
  - it's just a many-to-one relationship set from class to course
  - However, in reality, a class entity is still existence dependent on a course entity!
    - We CANNOT have a class based on a *non-existing course*! (TOTAL participation)

Note on Weak Entity Set

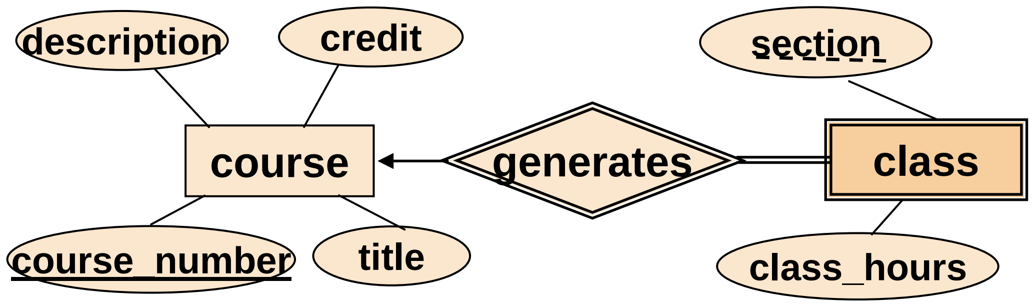
- So why do we even have the notion of weak entity then?
  - Realize that a weak entity can be turned into a strong entity by creating a primary key of its own
- Whether to adapt a weak entity set or adapt a strong entity set with a standard many-to-one relationship with a total participation largely depends on
  - Can we **create globally unique identification scheme** that can be used *as the primary key*?

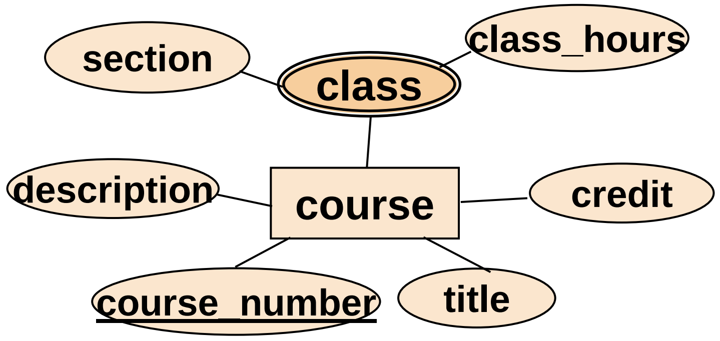
Weak Entity Set vs. Strong Entity Set - 5



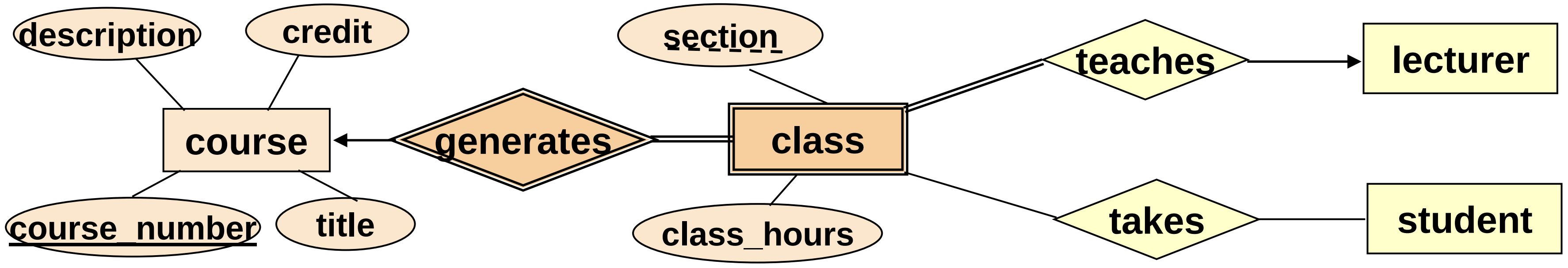
- Combination of **order_number** and **item_number** uniquely identifies a particular item in an order.
    - If we can **create a globally unique identifier** for the same purpose, then it can be used as the **primary key** of **orderItem**.
- In certain enterprise, creation of such globally unique identifier may NOT be POSSIBLE;
    - Therefore, it still requires the use of weak entity set

Weak Entity Set vs. Multi-valued Attributes - 1



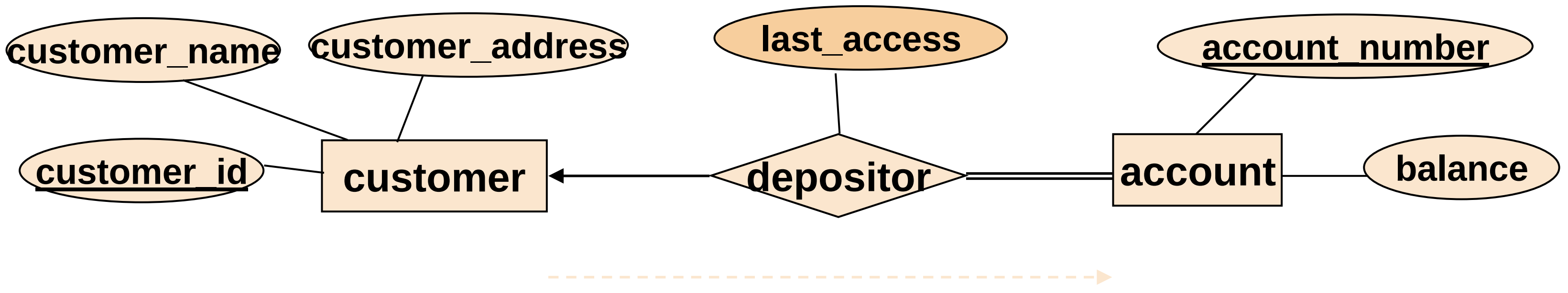


Weak Entity Set vs. Multi-valued Attributes - 2



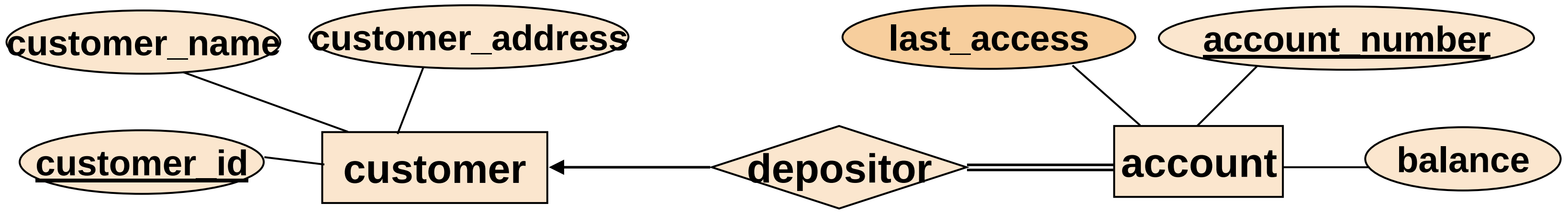
- The use of Weak Entity Set is preferred (necessary) *over multi-valued attribute approach* when the weak entity **needs to participate in other relationships** apart from the identifying relationship.

Placement of Descriptive Attributes - 1



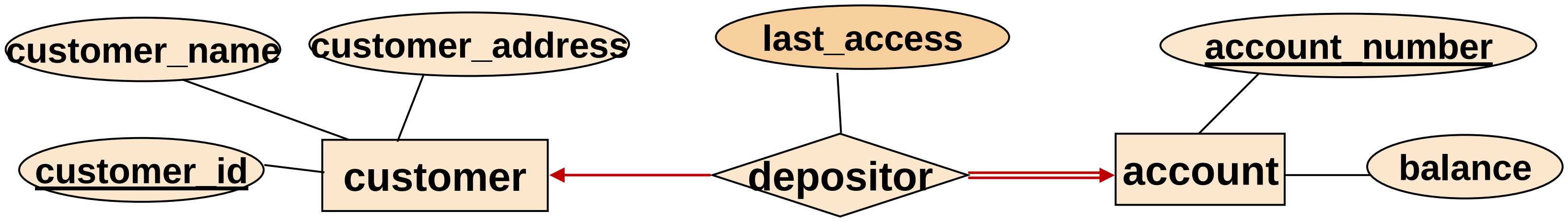
- One-to-Many (from customer to account)
  - A customer can have *many accounts*, **but** *no joint account holders* allowed
    - **last_access** is used to keep track of the last time a customer accessed an account

Placement of Descriptive Attributes - 2



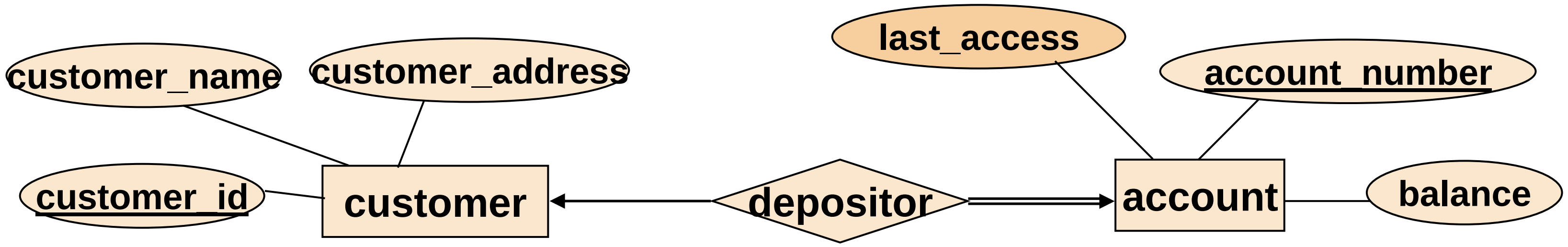
- Descriptive Attributes of One-to-Many (or Many-to-One) relationship sets **can be repositioned to** the entity set on the **'Many' side** of the relationship, especially, when the Many side is having **total participation.**

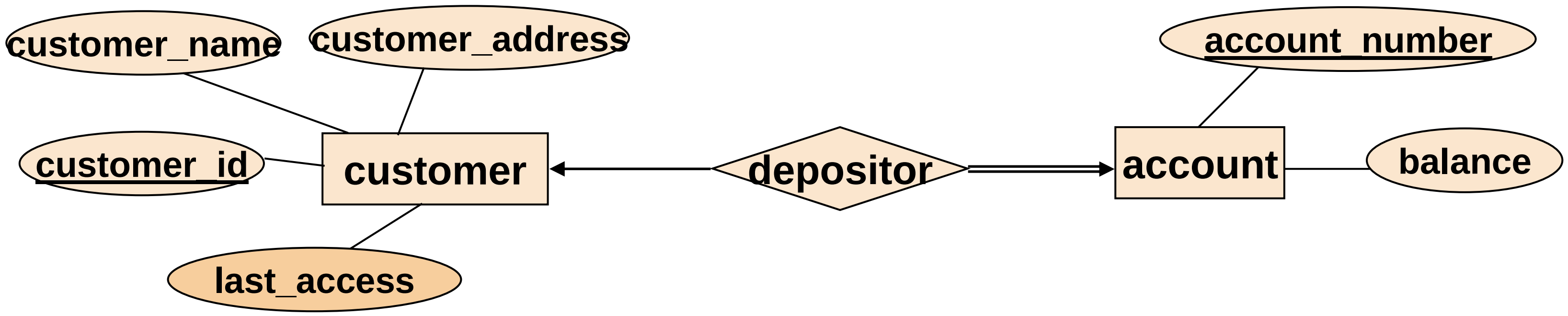
Placement of Descriptive Attributes - 3



- **One-to-One**
  - A customer can have at most one account
  - An account must be owned by one account

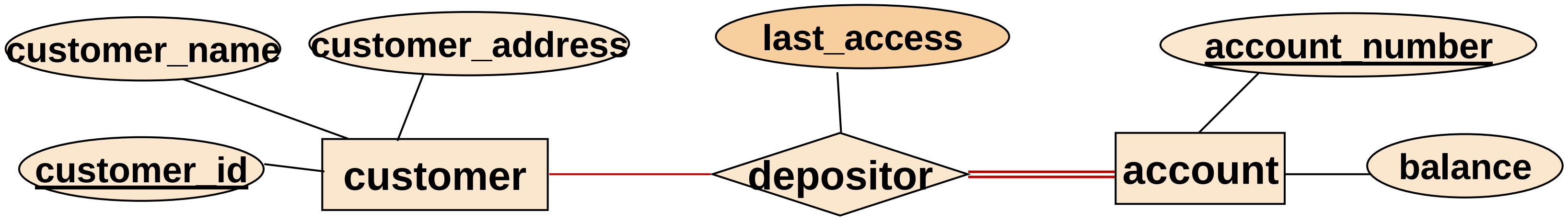
Placement of Descriptive Attributes - 3





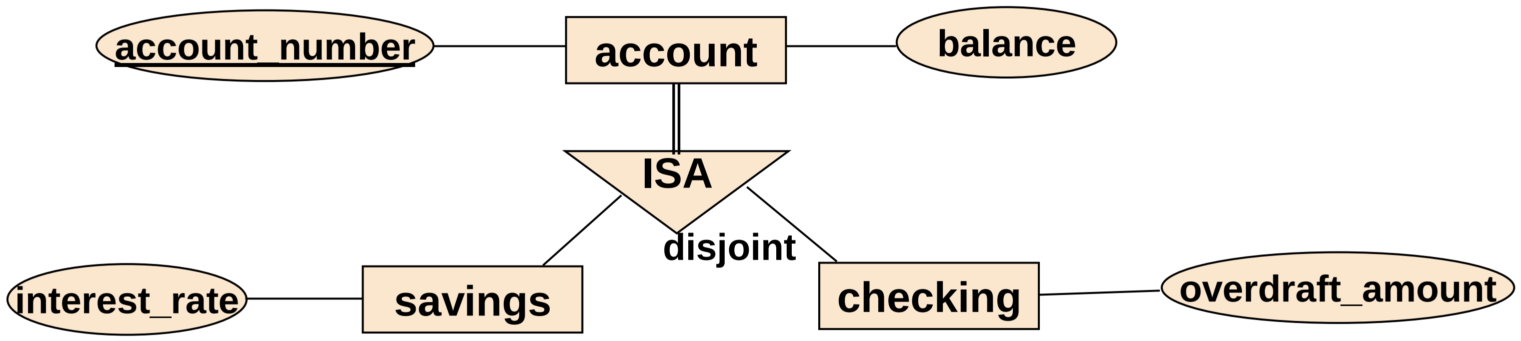
One-to-One

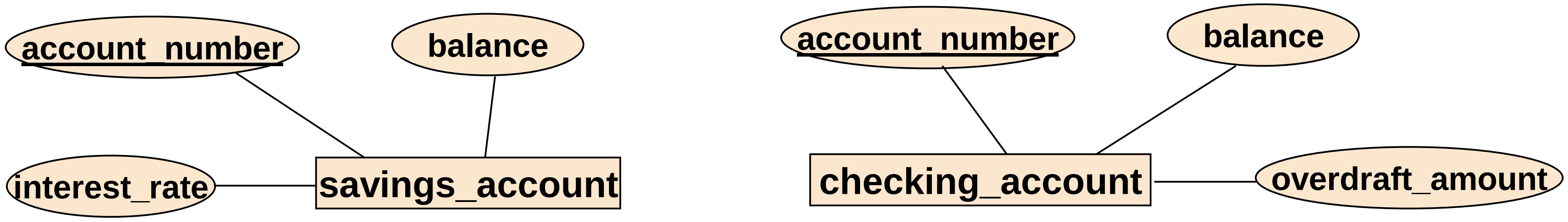
Placement of Descriptive Attributes - 4



- **Many-to-Many**
  - A customer can have zero, one, or more accounts
  - An account may be owned by many customers (join account holders)
  - Descriptive Attributes in Many-to-Many relationships **CANNOT be repositioned** to either entity sets
    - Not able to represent all the information

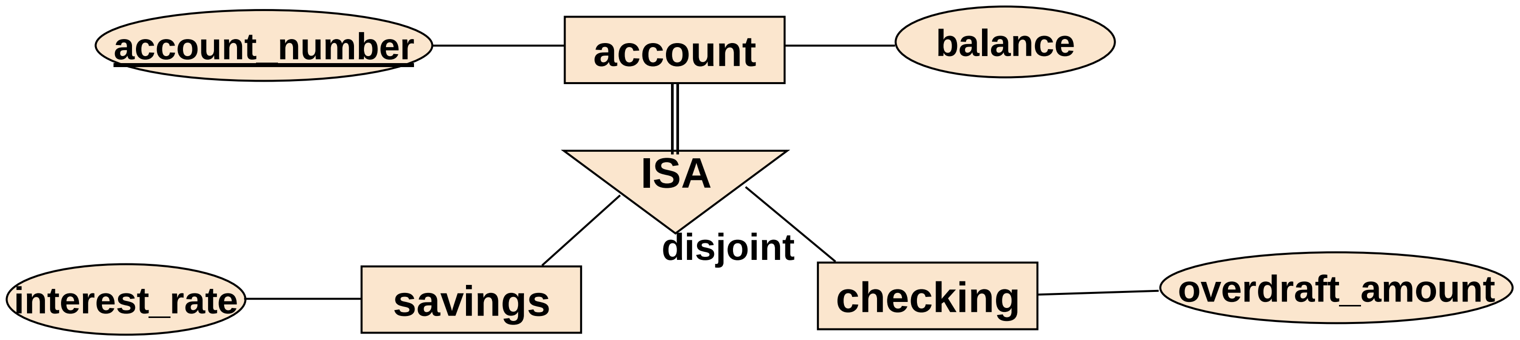
Inheritance Hierarchy vs. Separate Entity Sets





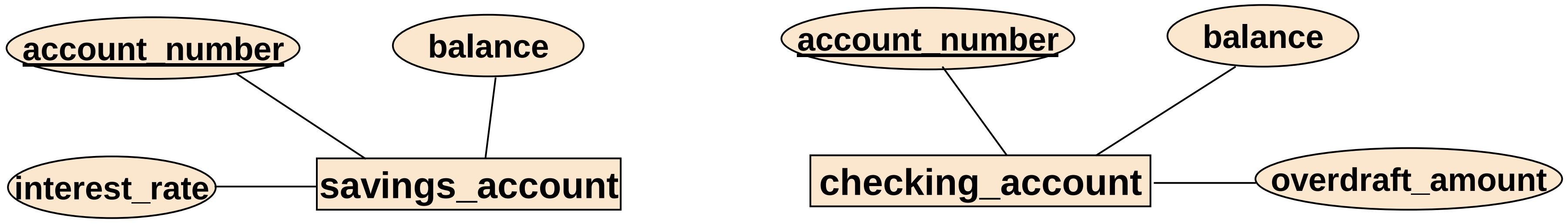
- How the two design approaches differ?

Inheritance Hierarchy



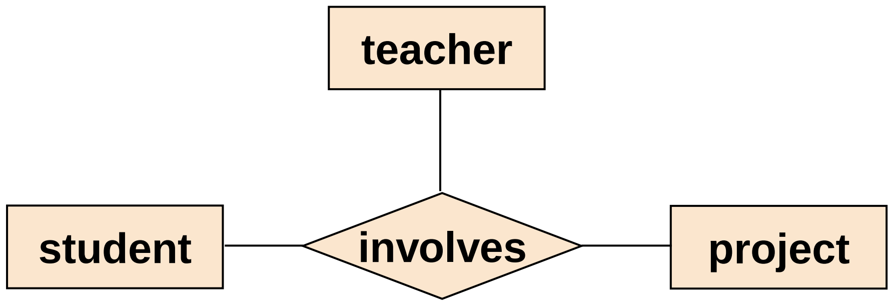
- Captures the semantics of **Specialization/Generalization Relationship**
  - *Savings* and *checking* accounts are special kind of accounts
  - Highlighted through **common shared attributes** in the superclass and **specific attributes** in each subclass
- Certain **Business rules** can be captured through **disjoint vs. overlapping** and **total vs. partial constraints**
- Simple to represent relationships with other entities that are common to all the entities in the hierarchy as well as subclass specific relationships that pertain only to certain groups of entities

Separate Entities



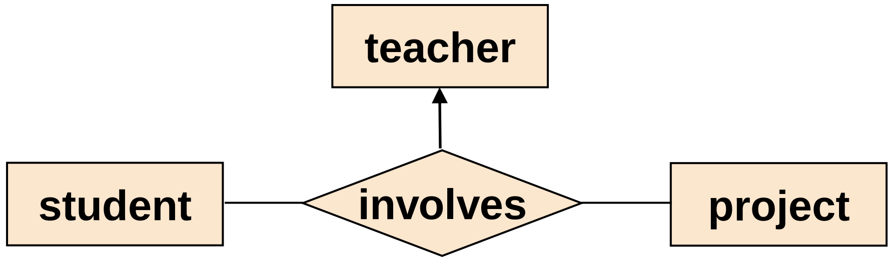
- Represents **two separate entity sets** (*possibly*) **unrelated** to each other
  - **Different Entity Sets can be related through Relationship sets**
    - However, Specialization/Generalization relationship is difficult or impossible to express using the standard entity-relationship model

N-ary Relationships vs. Multiple Binary Relationships



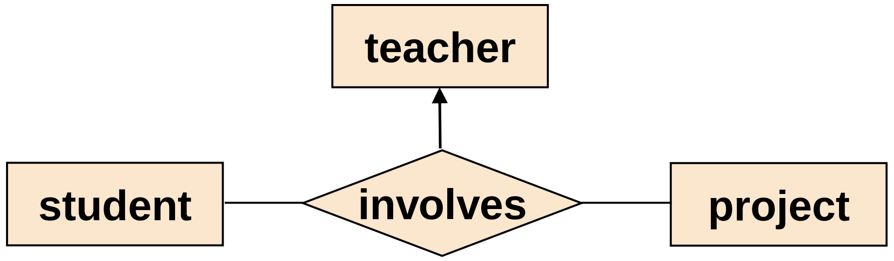
- Captures relationships among teachers, students and projects
  - students working on a particular project under supervision of a particular teacher
- *Participation may be partial*, but any given relationship instance will involve association of three entities: a student entity, a teacher entity and a project entity
- Notice that it's a (M-M-M) ternary relationship; What's the meaning?
  - A student can participate in **many projects** and can work under the supervision of **several teachers** for ANY OF these projects, and each teacher can supervise **many students** on ANY project.

Semantics of N-ary relationships - 1



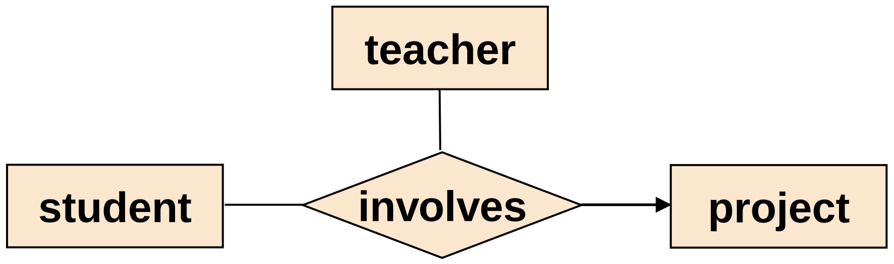
- Notice the arrow on the teacher entity set; (M-O-M); What's the meaning?
  - Each student can participate in many projects
  - Each student can work with many teachers
  - Each teacher can look after many projects
  - Each teacher can supervise many students
  - Each project can involve many students
  - Each project can involve many teachers

Semantics of N-ary relationships - 2



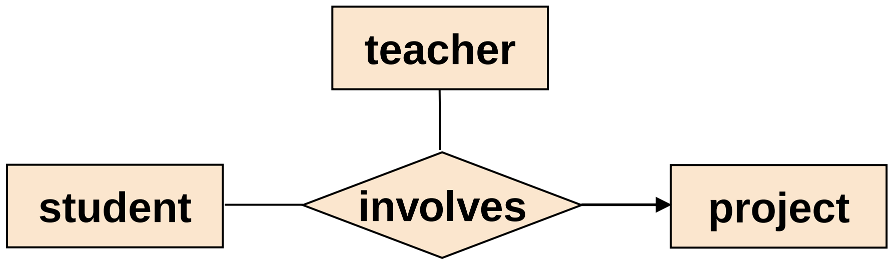
- A particular pair of a student entity and a teacher entity can participate in many projects
- A particular pair of a teacher entity and a project entity can involve many students
- **A particular pair of a student entity and a project entity can be associated with AT MOST ONE teacher entity**
  - For a particular project, a student can be supervised by at most one teacher.

Semantics of N-ary relationships - 3



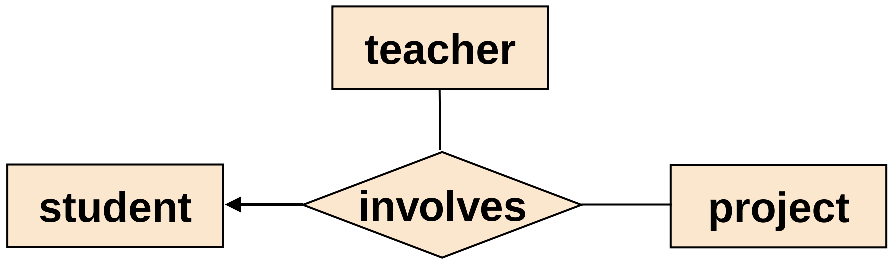
- Notice the arrow on the project entity set; (M-M-O); What's the meaning?
  - Each student can participate in many projects
  - Each student can work with many teachers
  - Each teacher can look after many projects
  - Each teacher can supervise many students
  - Each project can involve many students
  - Each project can involve many teachers

Semantics of N-ary relationships - 4



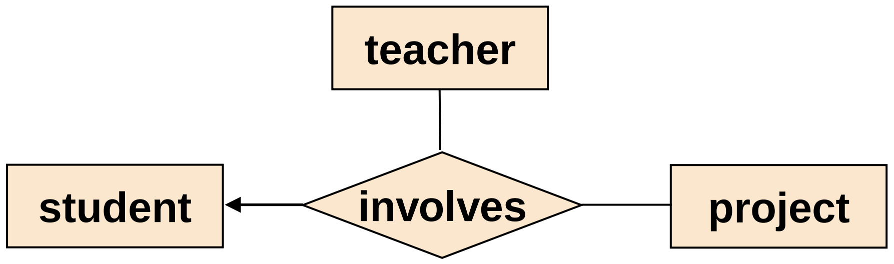
- A particular pair of a teacher entity and a project entity can involve many students
- A particular pair of a student entity and a project entity can involve many teachers
- **A particular pair of a student entity and a teacher entity can be associated with AT MOST ONE project entity**
    - A particular pair of a student and a teacher can work on at most one project

Semantics of N-ary relationships - 5



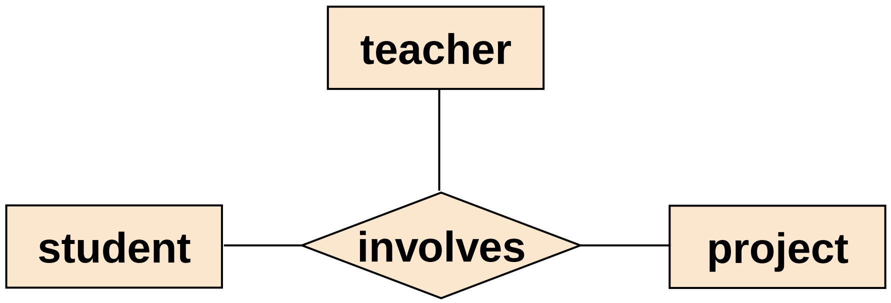
- Notice the arrow on the project entity set; (O-M-M); What's the meaning?
  - Each student can participate in many projects
  - Each student can work with many teachers
  - Each teacher can look after many projects
  - Each teacher can supervise many students
  - Each project can involve many students
  - Each project can involve many teachers

Semantics of N-ary relationships - 6

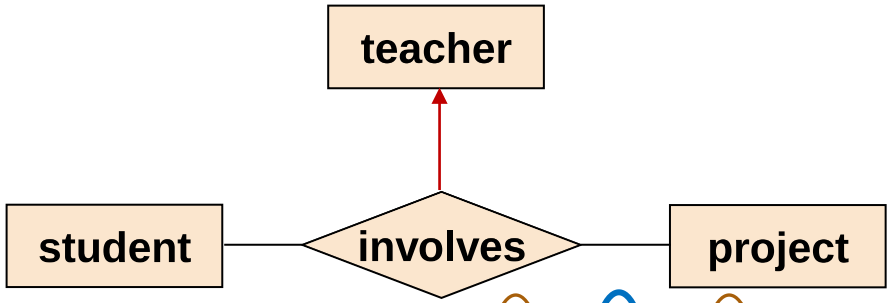


- A particular pair of a student entity and a project entity can involve many teachers
- A particular pair of a student entity and a teacher entity can involve many projects
- **A particular pair of a teacher entity and project entity can be associated with AT MOST ONE student entity**
  - A teacher running a project can have at most one student working on the project

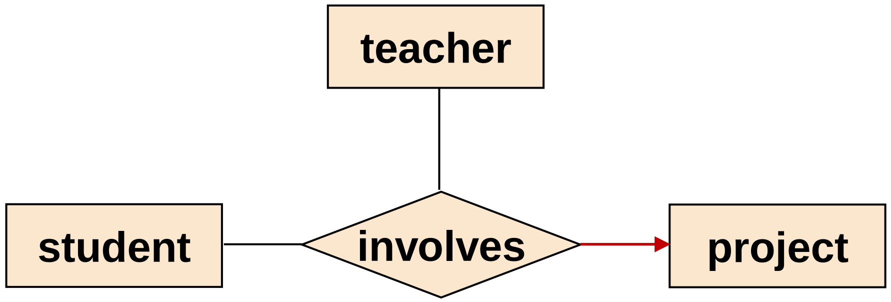
Semantics of N-ary relationships



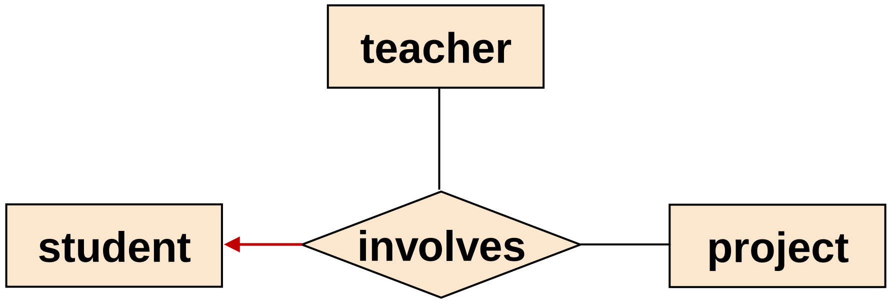
Assume the presence	S1	T1	P1
OK	S1	T2	P1
OK	S1	T1	P2
OK	S2	T1	P1



Assume the presence	S1	T1	P1
NOT ALLOWED	S1	T2	P1
OK	S1	T1	P2
OK	S2	T1	P1

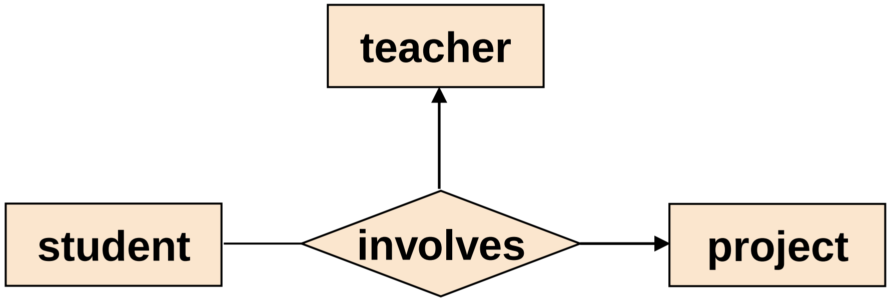


Assume the presence	S1	T1	P1
OK	S1	T2	P1
NOT ALLOWED	S1	T1	P2
OK	S2	T1	P1



Assume the presence	S1	T1	P1
OK	S1	T2	P1
OK	S1	T1	P2
NOT ALLOWED	S2	T1	P1

Semantics of N-ary relationships

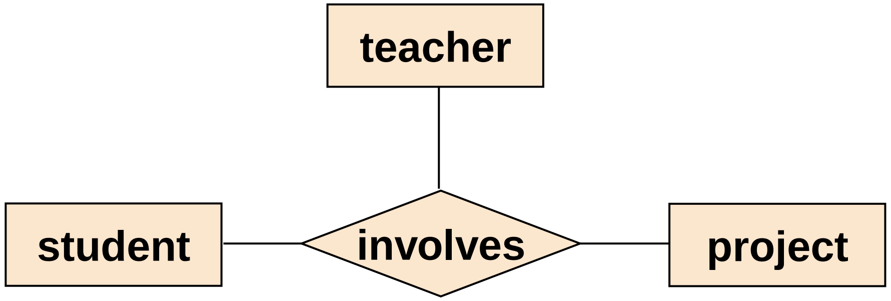


- **M-O-O (student, teacher, project); What's the meaning?**
- Interpretation 1: "A particular student entity can be associated with *at most one pair* of teacher and project entity"
  - In the presence of (S1, T1, P1), S1 cannot be associated with any other combination of teacher and project entities; e.g) (S1, T2, P2) NOT allowed
- Interpretation 2: "A particular student entity can be associated with *at most one teacher entity or one project entity*"
  - In the presence of (S1, T1, P1), S1 cannot be associated with T1 or P1, but S1 can be associated with other teacher and project entities
    - e.g.) (S1, T1, P2) not allowed
    - e.g.) (S1, T2, P1) not allowed
    - e.g.) (S1, T2, P2) ALLOWED
- Due to the ambiguity of the interpretation, we only allow **AT MOST ONE ARROW** in any N-ary relationships

N-ary vs. Multiple Binary Relationships - 1

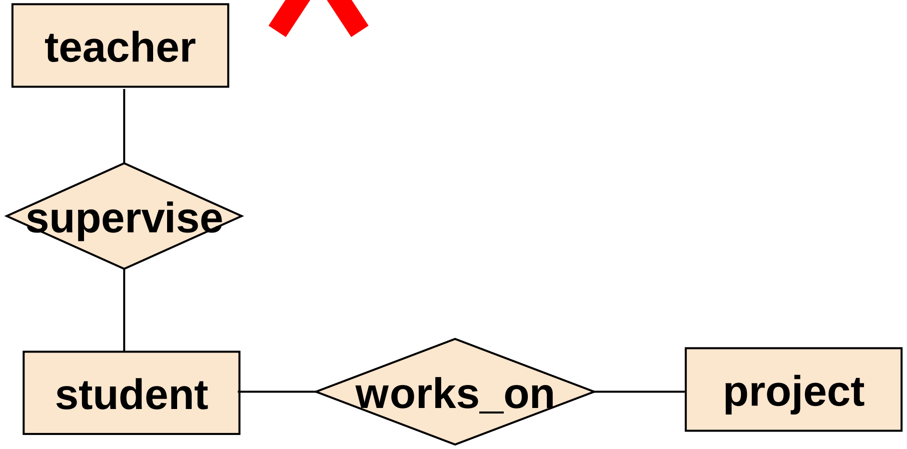
- **Many Design Tools (CASE tools) do NOT** support the concept of N-ary Relationships.
  - Furthermore, **some people encourages the disuse of** N-ary relationships and **sticking to only binary relationships** to simplify the design space.
- Can we **transform N-ary Relationships into multiple binary relationships** that are ***semantically equivalent*****?**

N-ary vs. Multiple Binary Relationships -2



teacher	student	project
T1	S1	P1
T1	S2	P2
T2	S1	P2

Choice 1 ✗



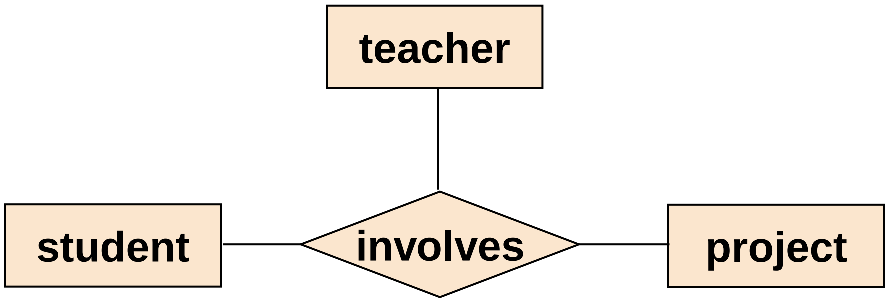
teacher	student
T1	S1
T1	S2
T2	S1

student	project
S1	P1
S1	P2
S2	P2

Spurious records

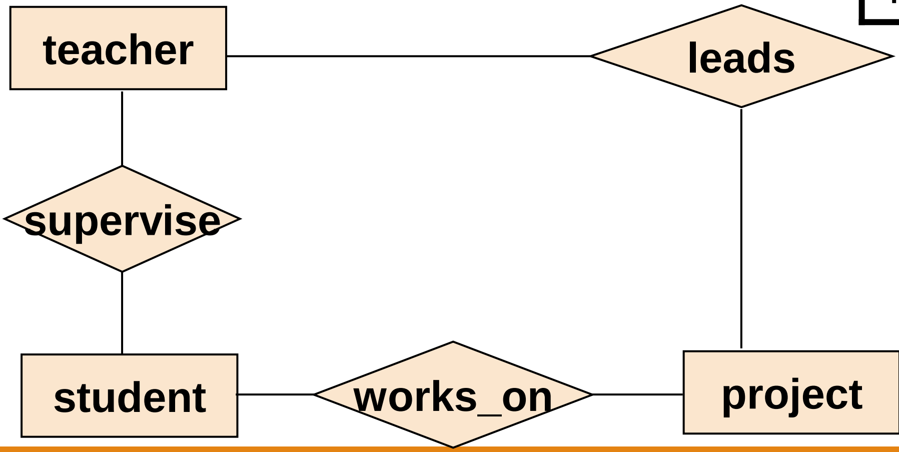
teacher	student	project
T1	S1	P1
T1	S1	P2
T1	S2	P2
T2	S1	P1
T2	S1	P2

N-ary vs. Multiple Binary Relationships



teacher	student	project
T1	S1	P1
T1	S2	P2
T2	S1	P2

Choice 2 ✗



teacher	student
T1	S1
T1	S2
T2	S1

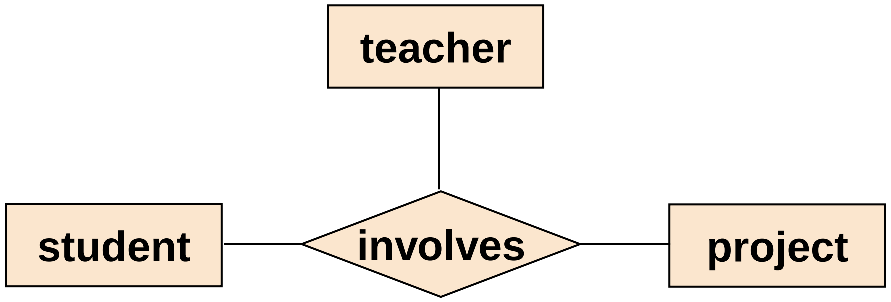
student	project
S1	P1
S1	P2
S2	P2

teacher	project
T1	P1
T1	P2
T2	P2

teacher	student	project
T1	S1	P1
T1	S1	P2
T1	S2	P2
T2	S1	P2

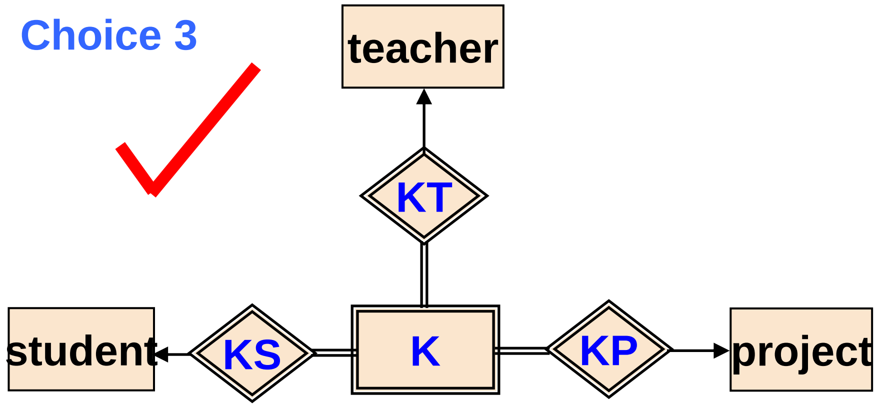
Spurious record

N-ary vs. Multiple Binary Relationships



teacher	student	project
T1	S1	P1
T1	S2	P2
T2	S1	P2

Choice 3



K
K1
K2
K3

K	teacher
K1	T1
K2	T1
K3	T2

K	student
K1	S1
K2	S2
K3	S1

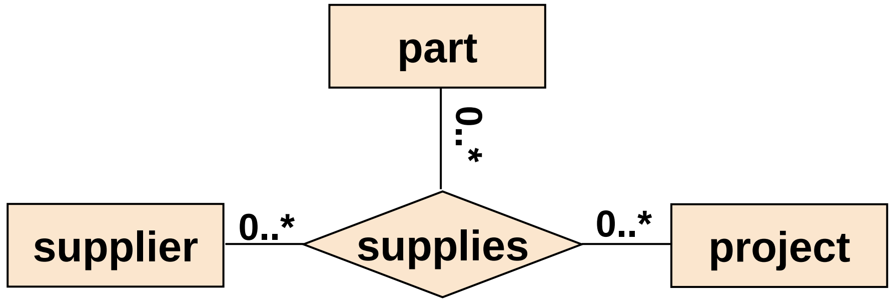
K	project
K1	P1
K2	P2
K3	P2

K
K1
K2
K3

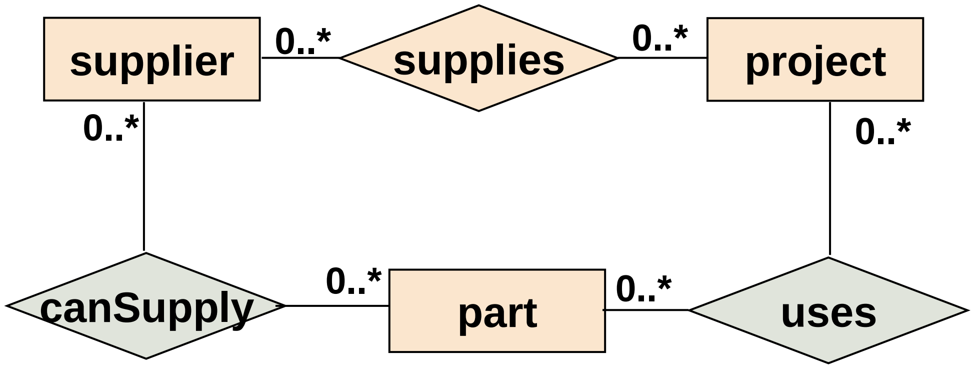
teacher	student	project
T1	S1	P1
T1	S2	P2
T2	S1	P2

- **Any N-ary relationships can be transformed into multiple binary identifying relationships, as demonstrated.**
  - Create **a weak entity set** whose identification depends on the combination of the entities involved in the original n-ary relationship
  - Link the weak entity to each of the entity sets in the original n-ary relationship

N-ary vs. Multiple Binary Relationships

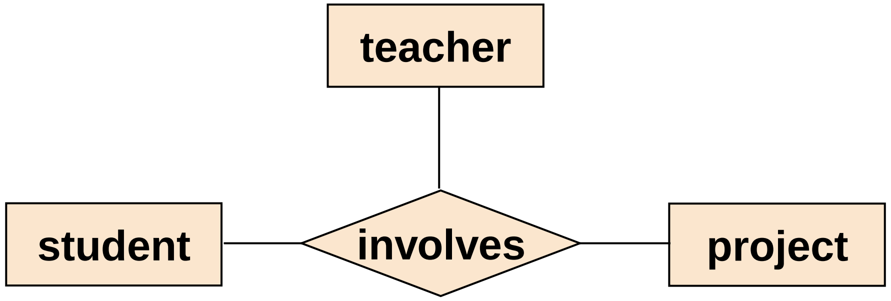


- The ternary relationship keeps track of which supplier supplies what parts to which project.
  - e.g.) "X-factor" provides "motors" to the project "genome"



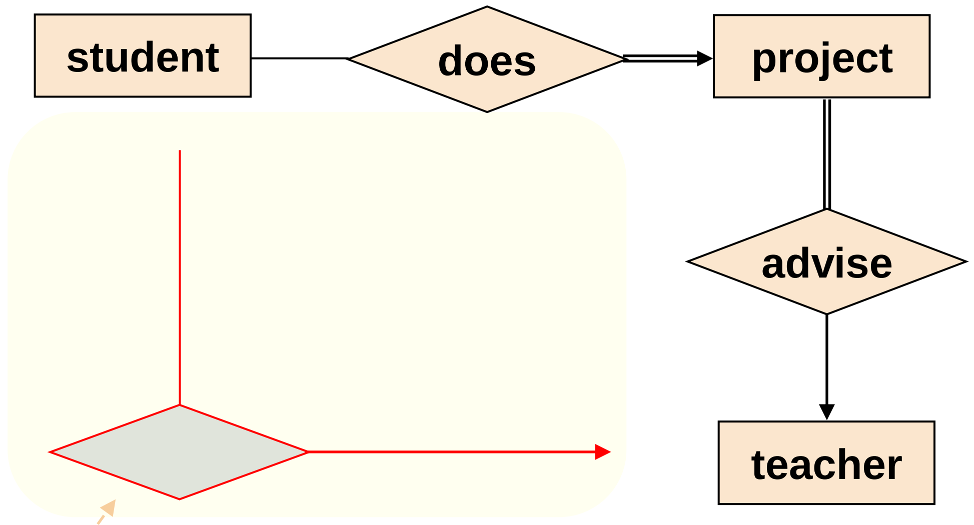
- As we saw earlier in the 2nd previous slide, they are **not semantically equivalent**
  - In three binary relationship version, certain information cannot be represented (may generate information which does not exist in the original ternary relationship)

N-ary vs. Multiple Binary Relationships



- Assume the following additional constraints
  - Student can do only a single project (senior project?)
  - Student can have only one teacher as a project advisor
  - A project include a teacher as an advisor and one or more students
  - A teacher can be advisors to different projects
  - There can be teachers who are not involved in any project advising
  - There are students who are not doing a project
- Is the ternary relationship good reflection of the real-world situation?
  - What are the problems in the ternary problem?
    - How can we constrain that a student can do at most a single project
    - How can we constrain that a student can have only one advisor?

N-ary vs. Multiple Binary Relationships



- Two binary relationships linking students to projects and linking projects to their advisor are a BETTER design than a ternary relationship.
- Note: **relationship** between student and teacher is **redundant**; can be **removed** since
  - A student can be associated with only one project
  - A student can be associated with only one teacher
  - A project MUST be advised by a single teacher
  - Therefore, if a student does a project, he must have an advisor.

N-ary vs. Multiple Binary Relationships

- Neither N-ary relationships nor Multiple binary relationships are always better than the other.
  - In certain cases, N-ary relationships may include 'extra' information that cannot be possible in the real-world

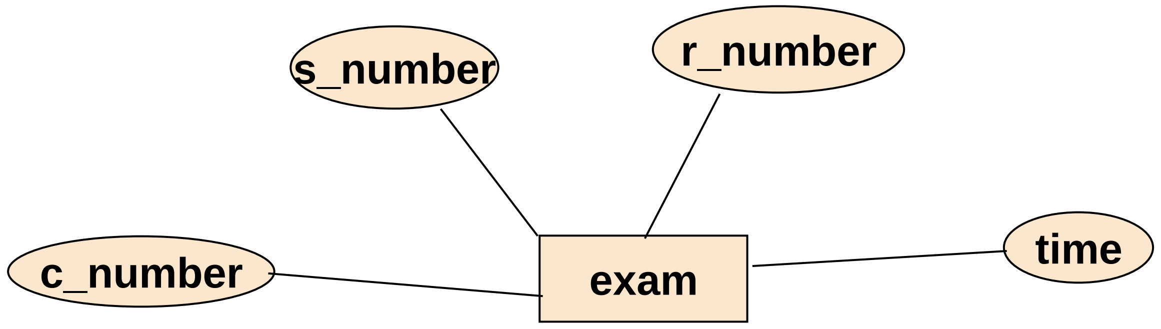
- It all depends on the semantics of the enterprise being modelled!
  - If **there are NO further cardinality constraints** between some of the entity sets within n-ary relationship, then N-ary relationship design may be a better reflection of the real world
  - If **there are cardinality constraints** between some of the entity sets within n-ary relationship, then the the real-world logic may be better represented by multiple binary relationships
  - In some cases, we may have to employee additional binary relationships among the entity sets on top of the n-ary relationship

- What if N-ary relationship is the BEST design (can't find other equivalence using only standard binary relationship) to reflect the real-world situation ***but*** you are NOT allowed to use N-ary relationship in your design (by your boss or your tool)?
  - You *can always transform any N-ary relationship into semantically equivalent* form by **creating a weak entity set** to replace the N-ary relationship and **by linking** the weak entity set **to other entity sets through identifying relationships**.

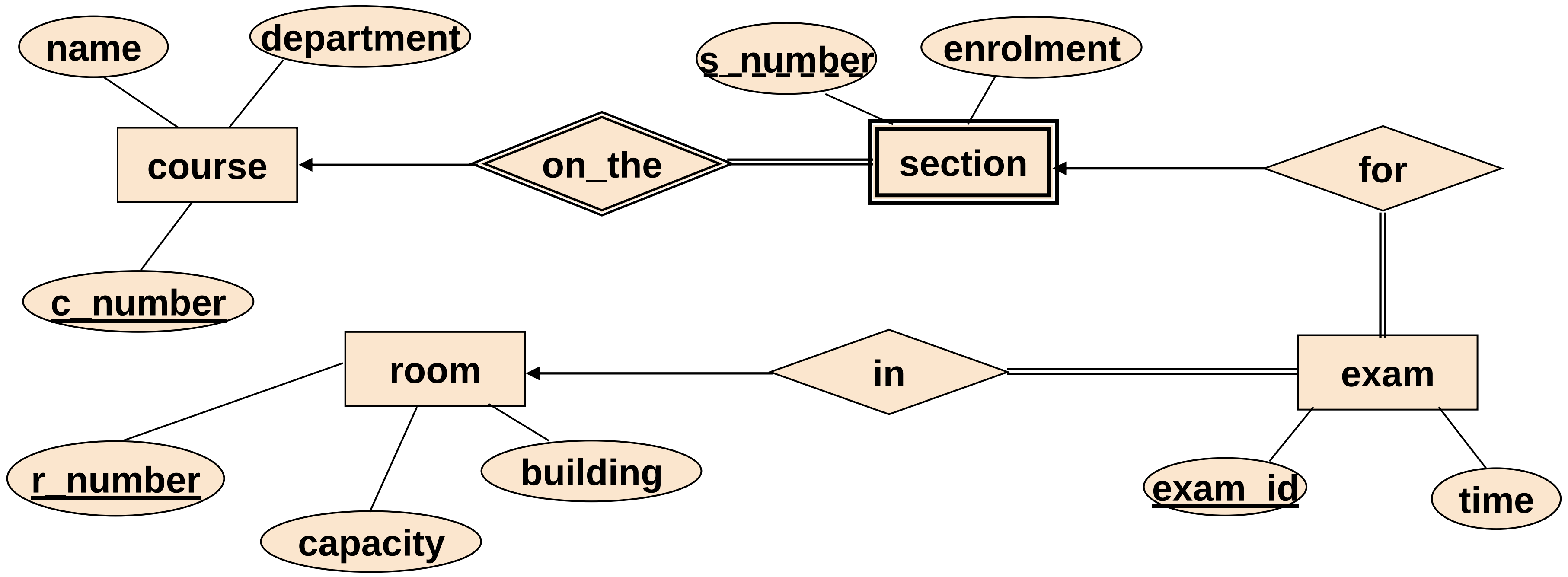
Self-Exercise #1

- Consider a university database for the scheduling of classrooms for final exams. This database could be modelled as the single entity set exam, with attributes course_number, section_number, room_number, and time. Alternatively, one or more additional entity sets could be defined, along with relationship sets to replace some of the attributes of the exam entity set, as
  - course with the following attributes
    - name, department and c_number
  - section, a weak entity set dependent on course
    - s_number, enrolment (the number of students taking the section)
  - Room with the following attributes
    - r_number, capacity, building
- Show an E-R diagram illustrating the use of all three additional entity sets listed
- Explain what application characteristics would influence a decision to include or not to include each of the additional entity sets.

Self-Exercise #1

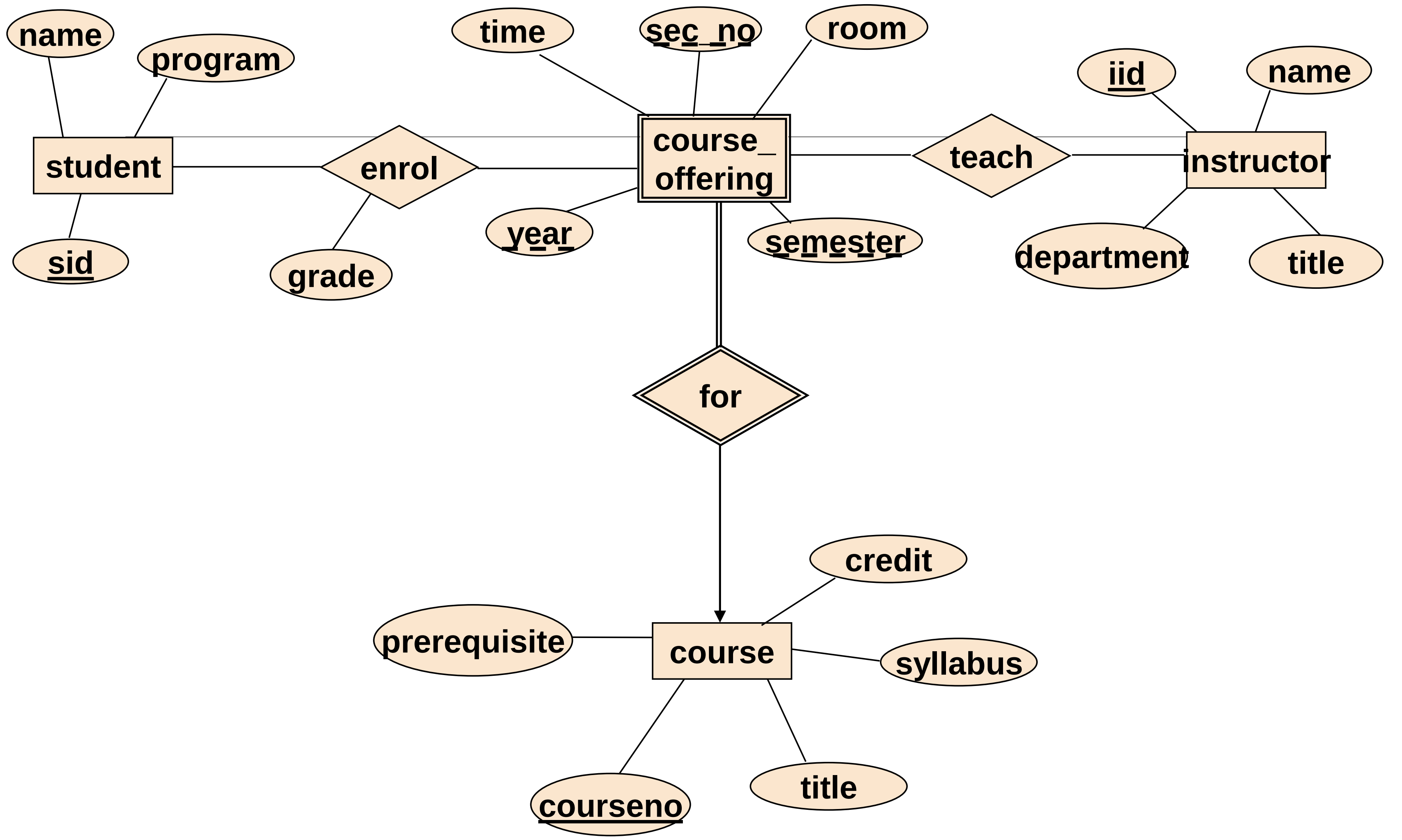


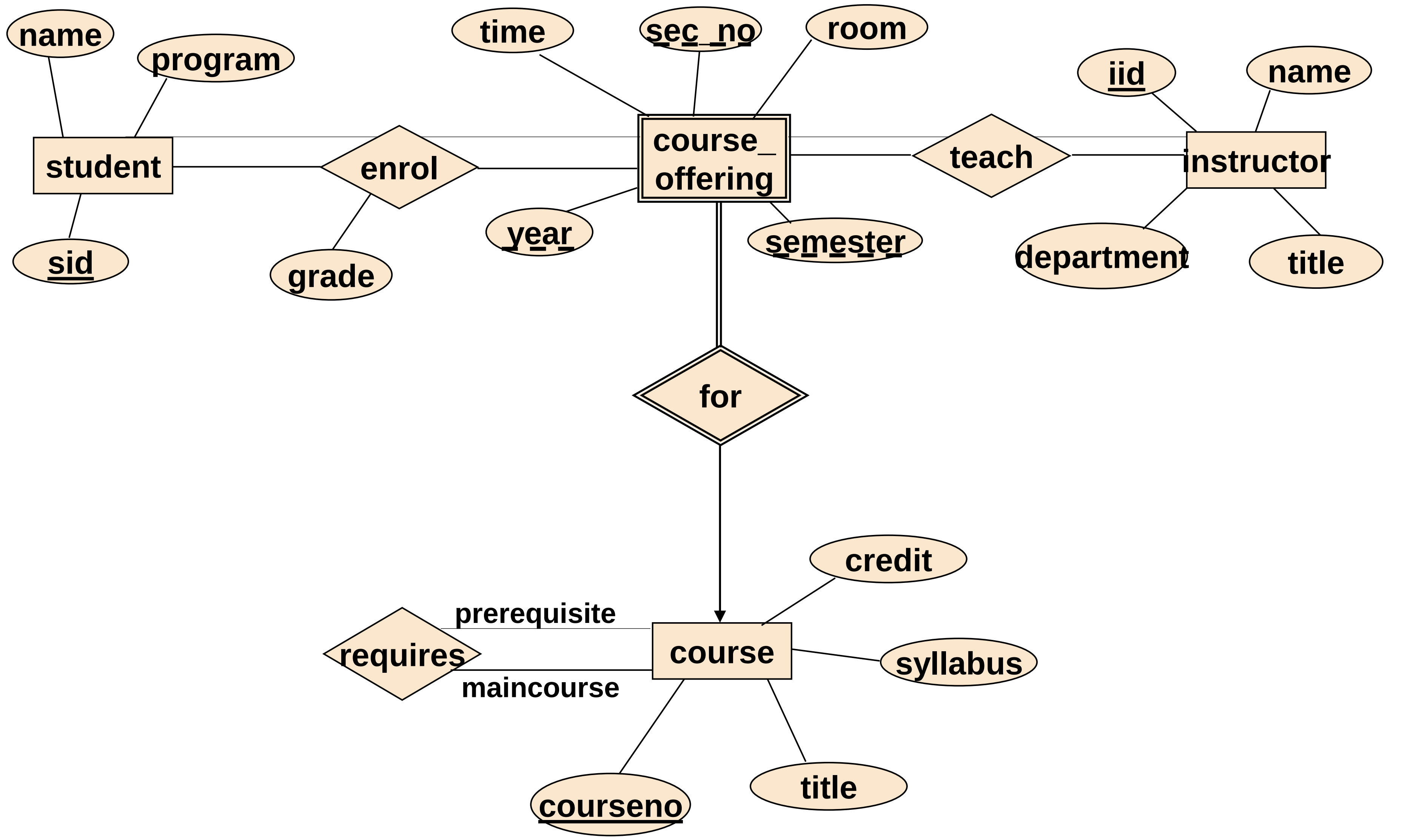
Self-Exercise #1

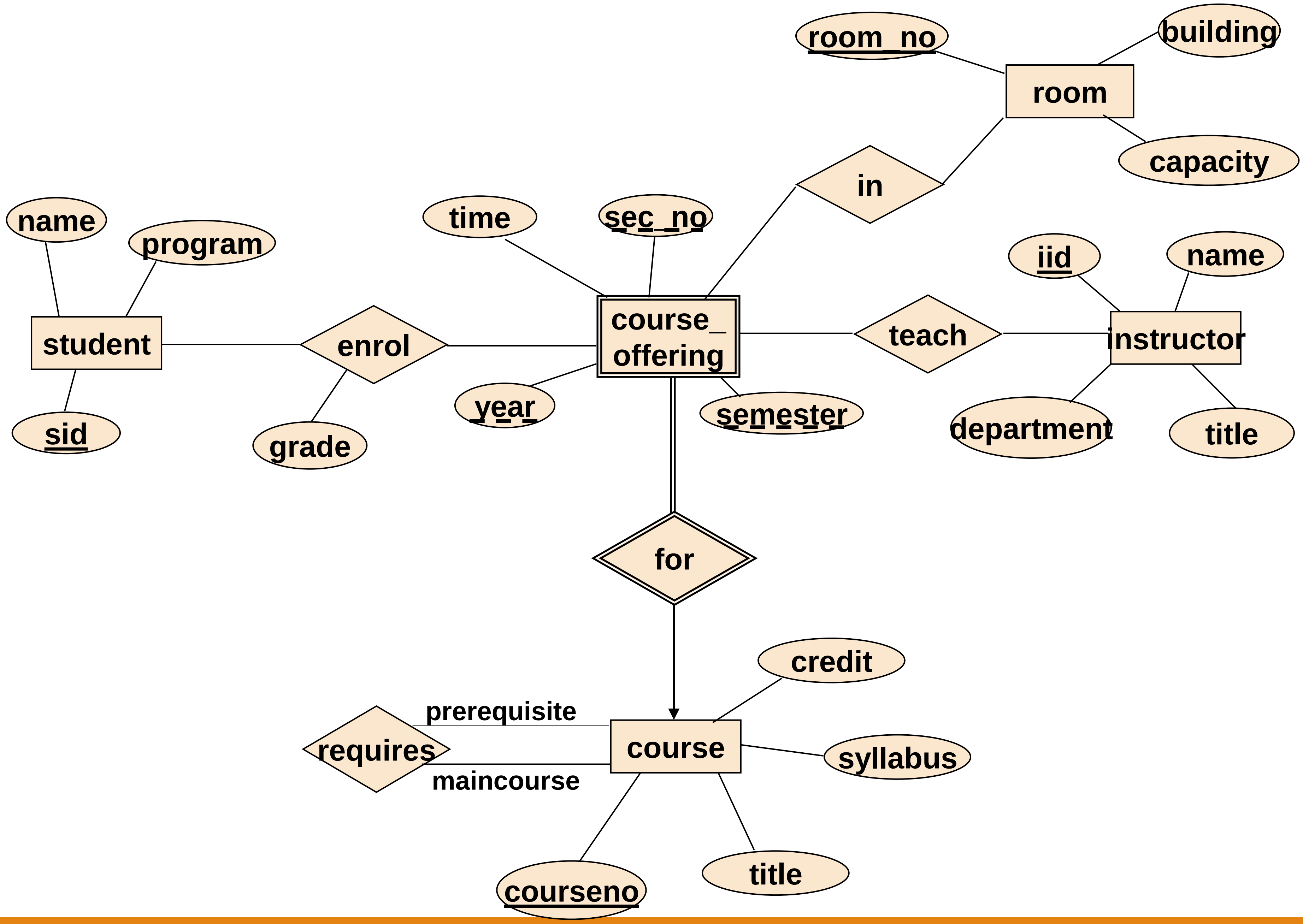


Self-Exercise #2

- A university registrar's office maintains data about the following entities:
  - course: number, title, credits, syllabus, prerequisites
  - course_offering: course number, year, semester, section number, instructor(s), classroom
  - student: student_id, name, program
  - instructor: identification number, name, department, title
- Furthermore, the enrolment of students in courses and grades awarded to students in each course they are enrolled for must be appropriately modelled.
- **Do the following strictly in the order given!**
  - Construct an E-R diagram for the registrar's office. Document all assumptions that you make, especially about the mapping cardinalities
  - Now construct at least one more E-R diagram which uses different design than the first one you made.
  - In the following slides, there are three alternative designs for the registrar's office. Compare them to your own designs and DISCUSS the differences, pros and cons of each alternative.
  - Try answering the design questions in the last slide and carry out the suggested self-research


name
program
student
sid
enrol
grade
time
sec_no
room
course_ offering
year
semester
teach
iid
name
instructor
department
title
for
credit
prerequisite
course
syllabus
courseno
title


name
program
student
sid
enrol
grade
time
sec_no
room
course_ offering
year
semester
teach
iid
name
instructor
department
title
for
credit
prerequisite
requires
maincourse
course
syllabus
courseno
title


room_no
building
room
capacity
in
name
program
time
sec_no
iid
name
student
enrol
course_
offering
teach
instructor
sid
grade
year
semester
department
title
for
credit
prerequisite
requires
course
syllabus
maincourse
courseno
title

Self-Exercise #2 (Cont.)

- Can the designs (including yours and the three shown previously) represent classes that meet at different places at different times?
  - e.g.) section 51 of SC2212 meets twice a week
    - First session from 9:30 till 11:00 on Mondays at E.34
    - Second session from 15:30 till 17:00 on Thursdays at E.41
- How would you modify the designs to model classes meeting at different places at different times?
- Can the designs guarantee that the database does not have two classes meeting at the same place and time?
  - How would you modify the designs to guarantee that the database does not allow conflicting scheduling of classes?
    - Section 51 of SC2212 at E.33 from 8:00 till 11:00
    - Section 53 of SC2282 at E.33 from 9:30 till 12:30
  - Do you think such business rules can be enforced automatically via DBMS data integrity checking? If yes, how? If no, why not?
    - What mechanisms in modern DBMS can be used to enforce complex business rules like this?
      - Carry out self-research on the following topics:
        - *Triggers, Stored Procedures (PL/SQL, T-SQL)*
        - *Three-tier Architecture: Presentation | (Application) Business Logic | Data*