Chapter 5 Entity-Relationship Model

Goliath Li
United International College

- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

Modeling

- A database can be modeled as:
  - a collection of entities and
  - relationship among entities.
- An entity is an object that exists and is distinguishable from other objects.
  Example: specific person, comany, event, or plant
- An entity set is a set of entities of the same type that share the same properties.
  Example: set of all persons, companies, trees, holidays

Entity Sets *customer* and *loan*

c_id	c_name	c_street	c_city
321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rey
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

customer

loan_num	amount
L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

Relationship Sets

- A relationship is an association among several entities. Example:

Hayes	depositor	A-102
custormer entity	*relationship* set	*account* entity

- A relationship set is a mathematical relation among $n \geq 2$ entities, each taken from entity sets

$$\{(e_1, e_2, \cdots, e_n) | e_1 \in E_1, e_2 \in E_2, \cdots, e_n \in E_n\},$$

where $(e_1, e_2, \cdots, e_n)$ is a relationship.
Example: $(Hayes, A - 102) \in depositor$

Relationship Set *borrower*

321-12-3123	Jones	Main	Harrison
019-28-3746	Smith	North	Rye
677-89-9011	Hayes	Main	Harrison
555-55-5555	Jackson	Dupont	Woodside
244-66-8800	Curry	North	Rey
963-96-3963	Williams	Nassau	Princeton
335-57-7991	Adams	Spring	Pittsfield

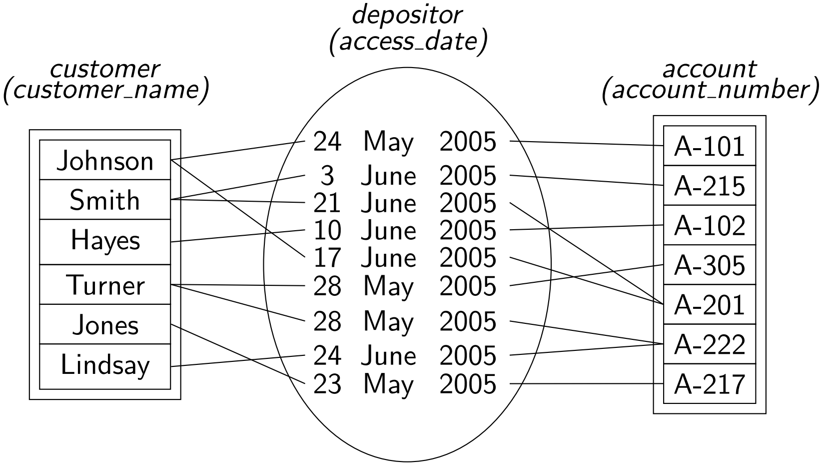
customer

L-17	1000
L-23	2000
L-15	1500
L-14	1500
L-19	500
L-11	900
L-16	1300

loan

Relationship Sets (Cont.)

- An attribute can also be property of a relationship set.
- For instance, the *depositor* relationship set between entity sets *customer* and *account* may have the attribute *access-date*.



Degree of a Relationship Set

- Refers to number of entity sets that participate in a relationship set.
- Relationship sets that involve two entity sets are binary (or degree two). Generally, most relationship sets in a database system are binary.
- Relationship sets may involve more than two entity sets.
  - Example: Suppose employees of a bank may have jobs (responsibilities) at multiple branches, with different jobs at different branches. Then there is a ternary relationship set between entity sets *employee*, *job*, and *branch*.
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)

Attributes

- An entity is represented by a set of attributes, that is descriptive properties possessed by all members of an entity set.
  Example:

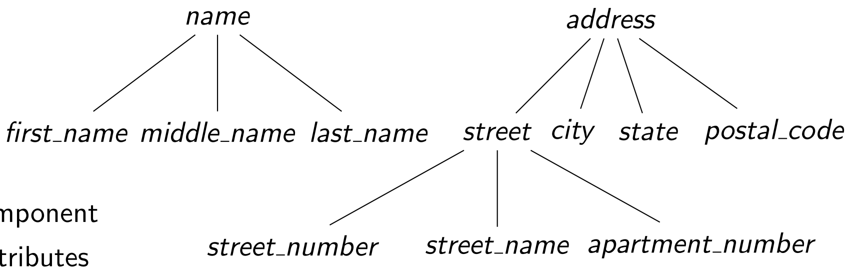
$$\begin{aligned} customer =& (customer\_id,\ customer\_name, \\ & customer\_street,\ customer\_city) \\ loan =& (loan\_number,\ amount) \end{aligned}$$

- Domain – the set of permitted values for each attribute
- Attribute types:
  - *Simple* and *composite* attributes.
  - *Single-valued* and *multi-valued* attributes
    Example: multivalued attribute: *phone_numbers*
  - *Derived* attributes
    Can be computed from other attributes
    Example: *age*, given *date_of_birth*

Composite Attributes

Composite

Attributes



Component

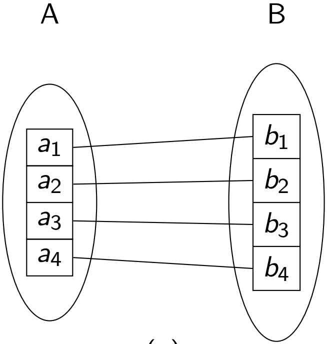
Attributes

- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

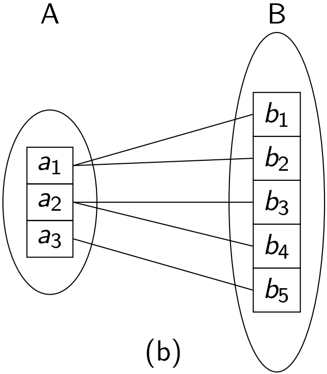
Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.
- Most useful in describing binary relationship sets.
- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one
  - One to many
  - Many to one
  - Many to many

Mapping Cardinalities



(a)



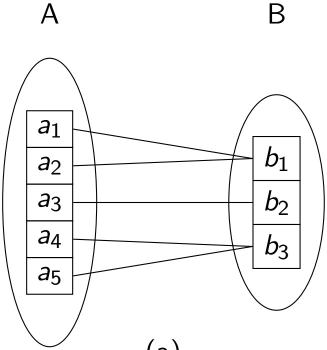
(b)

One to one

One to many

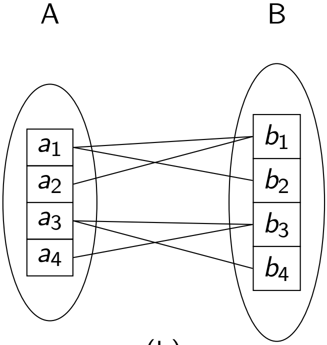
Note: Some elements in A and B may not be mapped to any elements in the other set.

Mapping Cardinalities



(a)

Many to one



(b)

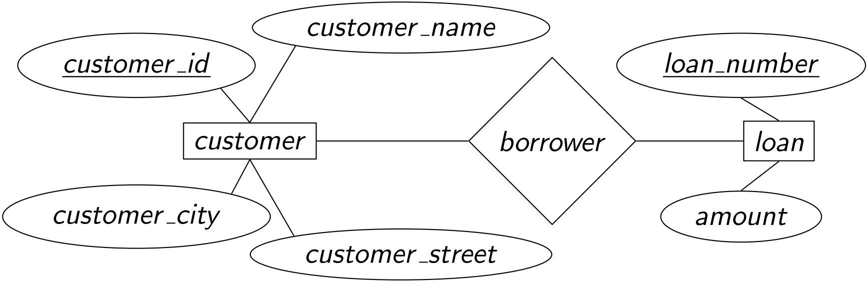
Many to many

Note: Some elements in A and B may not be mapped to any elements in the other set.

- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

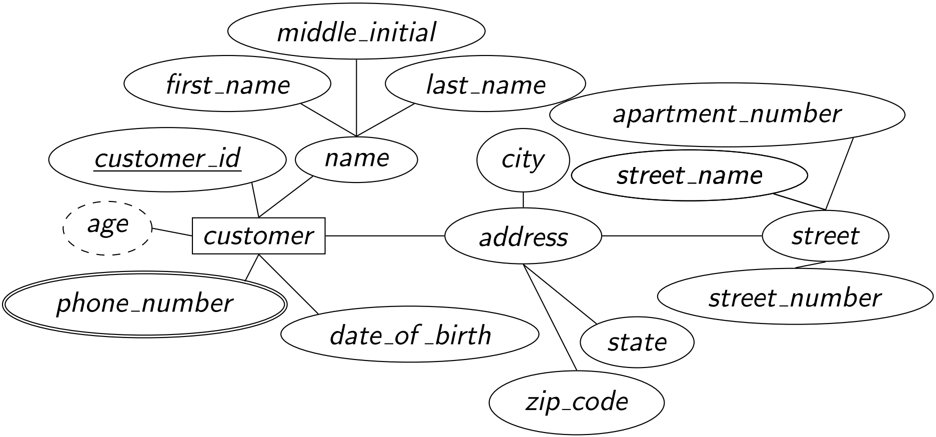
E-R Diagrams

- **Rectangles** represent entity sets.
- **Diamonds** represent relationship sets.
- **Lines link** attributes to entity sets and entity sets to relationship sets.
- **Ellipses** represent attributes.
- **Underline** indicates primary key attributes (will study later).



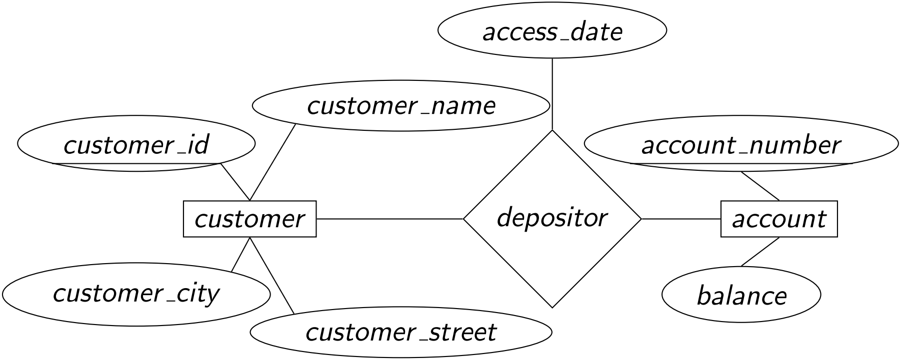
Composite, Multivalued, and Derived Attributes

- **Double ellipses** represent multivalued attributes.
- **Dashed ellipses** denote derived attributes.



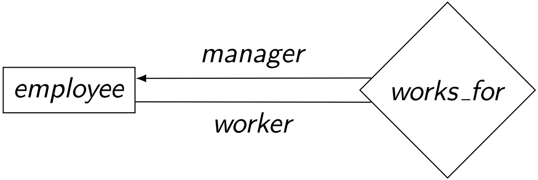
Relationship Sets with Attributes

Sometimes relationship sets may have attributes.



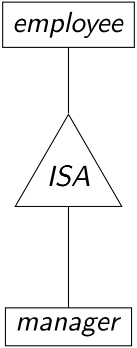
Roles

- Entity sets of a relationship need not be distinct.
- The labels *"manager"* and *"worker"* are called roles. They specify how employee entities interact via the *works_for* relationship set.
- Roles are indicated in E-R diagrams by labeling the lines that connect diamonds to rectangles.
- Role labels are optional, and are used to clarify semantics of the relationship.



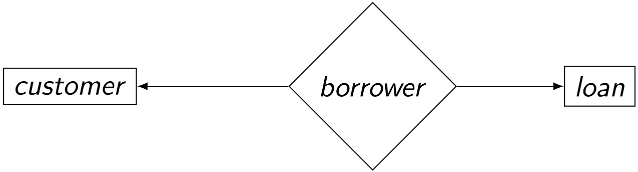
IS-A Relationship

- To show the relationship between two entity sets such that one is the subset of the other.
- IS-A relationships are presented by trangles.
- For example, all managers are employees. If we consider entity *employee* and *manager*, IS-A relationship connects the two entities.



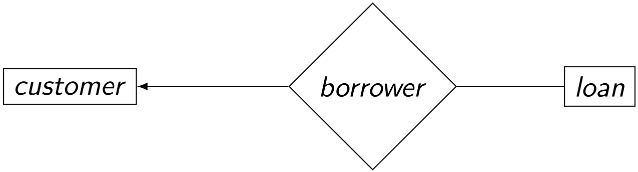
Cardinality Constraints

- We express cardinality constraints by drawing either a directed line ($\rightarrow$), signifying "one", or an undirected line ($-$), signifying "many", between the relationship set and the entity set.
- One-to-one relationship:
    - a customer is associated with at most one loan via the relationship borrower;
    - a loan is associated with at most one customer via borrower.



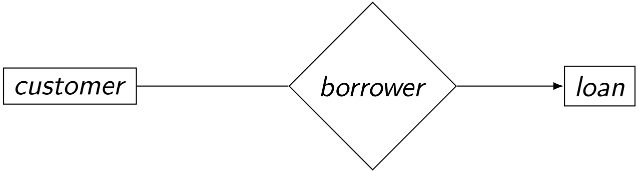
One-To-Many Relationship

- In the one-to-many relationship, a loan is associated with at most one customer via *borrower*, while a customer is associated with multiple (0 or many) loans via *borrower*.



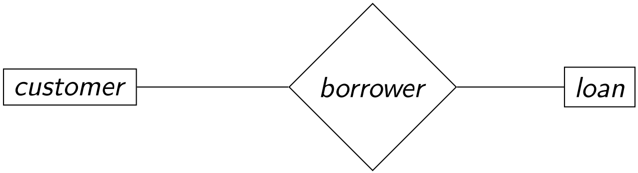
Many-To-One Relationship

- In a many-to-one relationship, a loan is associated with multiple (0 or many) customers via *borrower*, while a customer is associated with at most one loan via *borrower*.



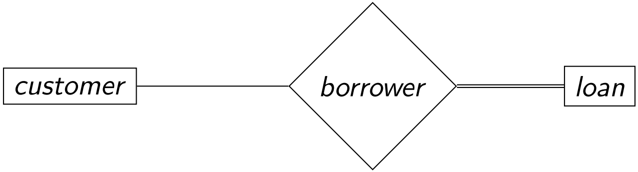
Many-To-Many Relationship

- In a many-to-one relationship, a loan is associated with multiple (0 or many) customers via *borrower*, while a customer is also associated with multiple (0 or many) loans via *borrower*.



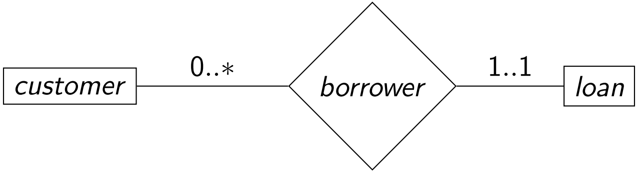
Participation of an Entity Set in a Relationship Set

- Total participation (indicated by double line): every entity in the entity set participates in at least one relationship in the relationship set.
  For example, the participation of *loan* in *borrower* is total.
  Every loan must have a customer associated to it via *borrower*.
- Partial participation: some entities may not participate in any relationship in the relationship set.
  For example, the participation of *customer* in *borrower* is partical.



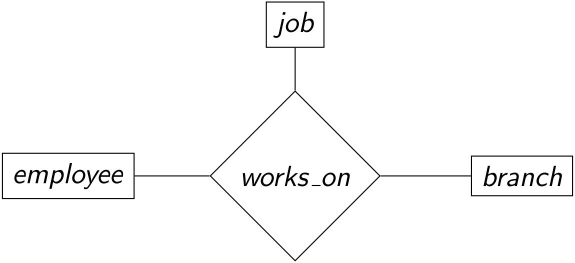
Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints.



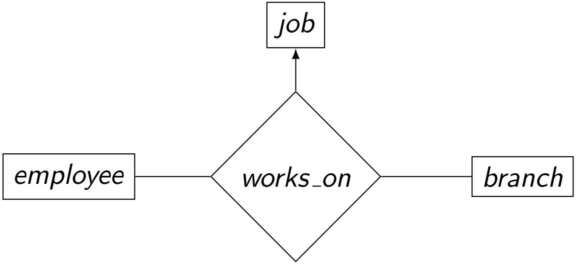
E-R Diagram with a Ternary Relationship

- Ternary (from Latin ternarius) means the composition of three items.
- Ternary relationship means three entities participate in one relationship.



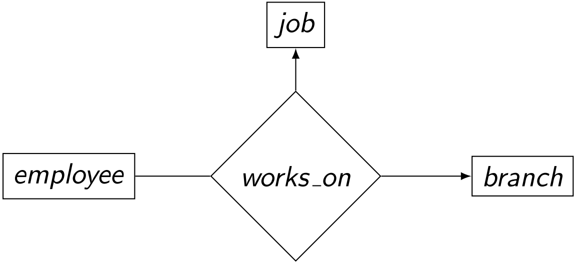
Cardinality Constraints on Ternary Relationship

- Cardinality constraints can also be applied on ternary relationships.
- For example, an arrow from *works_on* to *job* indicates each employee works on at most one job at any branch.



Cardinality Constraints on Ternary Relationship

- If there are multiple arrows, the diagram is ambiguous.



1. each employee works in at most one branch **and** on at most one job;
2. each employee works on at most one job but in multiple branches; or
3. each employee works in at most one branch but on multiple jobs.

- To avoid ambiguous, we only allow at most one arrow out of a ternary (or *n*-ary) relationship for cardinality constraints.

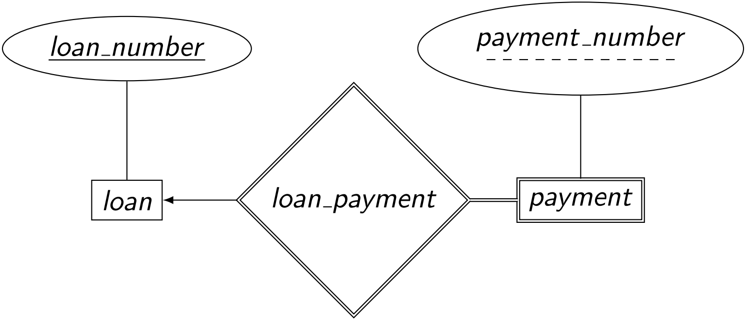
- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

Weak Entity Sets

- An entity set that does not have a primary key is referred to as a weak entity set.
- The existence of a weak entity set depends on the existence of a identifying entity set.
  - A weak entity set must relate to an identifying entity set via a total and one-to-many relationship set from the identifying to the weak entity set.
  - An identifying relationship is depicted by a double diamond.
- The discriminator (or *partial key*) of a weak entity set is the set of attributes that distinguishes among all the entities of a weak entity set.
- The primary key of a weak entity set is formed by the primary key of the strong entity set on which the weak entity set is existence dependent, plus the weak entity sets discriminator.

Weak Entity Sets (Cont.)

- **Double rectangles** depict weak entity sets.
- **Dashed lines** depict the discriminator of a weak entity set.
- For example, *payment_number* is the discriminator of *payment* entity set.
- The primary key for *payment* is *(loan_number, payment_number)*.



Weak Entity Sets (Cont.)

- A weak entity set can participate in relationships other than the identifying relationship. For example,
  - the *payment* entity could participate in a relationship with the *account* entity set, identifying the account from which the payment was made.
- A weak entity set may participate as owner in an identifying relationship with another weak entity set.
- A weak entity set is also possible to have a weak entity set with more than one identifying entity set.

More Weak Entity Set Examples

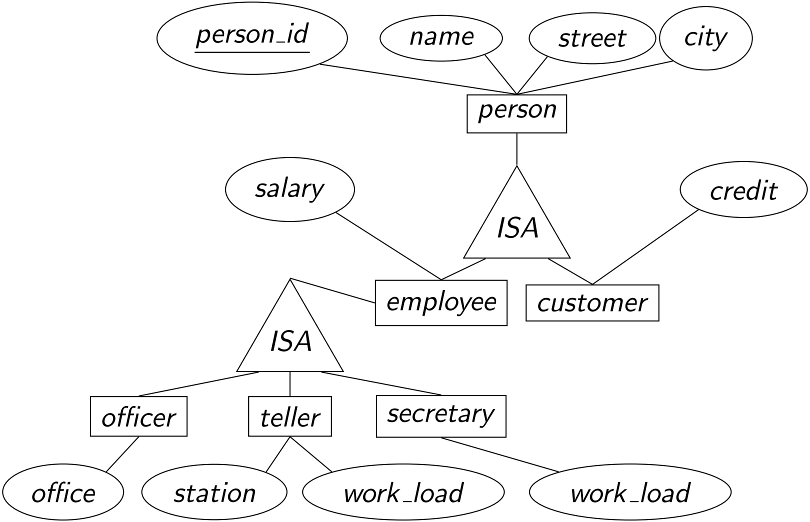
- In the database of universities, *course* is a strong entity while *course_offering* can be modeled as a weak entity.
- The discriminator of *course_offering* would be *semester* (including year) and *section_number* (if there are more than one section).
- If we model *course_offering* as a strong entity, *course_number* as an attribute. Then the relationship with *course* would be implicit in attribute *course_number*.

- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

Specialization

- Top-down design process; we designate subgroupings within an entity set that are distinctive from other entities in the set.
- These subgroupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a *triangle* component labeled ISA (E.g. customer is a person).
- **Attribute inheritance** a lower-level entity set inherits all the attributes and relationship participation of the higher-level entity set to which it is linked.

Specialization Example



Generalization

- **A bottom-up design process** combine a number of entity sets that share the same features into a higher-level entity set.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

Specialization vs Generalization

- Can have multiple specializations of an entity set based on different features.
- E.g. *permanent_employee* vs. *temporary_employee*, in addition to *officer* vs. *secretary* vs. *teller*
- Each particular employee would be
  - a member of one of *permanent_employee* or *temporary_employee*,
  - and also a member of one of officer, secretary, or teller.
- The ISA relationship also referred to as **superclass - subclass** relationship.

Constraints on specialization/generalization

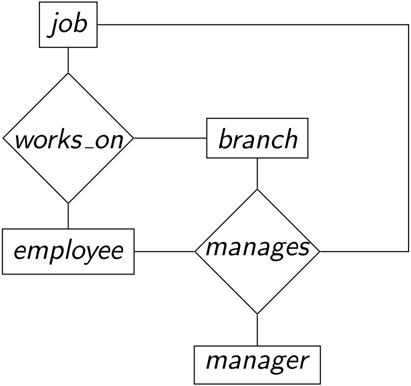
- Constraint on which entities can be members of a given lower-level entity set.
  - condition-defined
    Example: All customers over 65 years are members of senior-citizen entity set. A senior-citizen ISA a person.
  - user-defined
- Constraint on whether or not entities may belong to more than one lower-level entity set within a single generalization.
  - Disjoint
    An entity belongs to only one lower-level entity set.
    Noted in E-R diagram by writing disjoint next to the ISA triangle.
  - Overlapping
    An entity can belong to more than one lower-level entity sets.

Constraints on specialization/generalization

- **Completeness constraint** – specifies whether or not an entity in the higher-level entity set must belong to at least one of the lower-level entity sets within a generalization.
  - **total** : an entity must belong to one of the lower-level entity sets.
  - **partial**: an entity need not belong to one of the lower-level entity sets

Aggregation

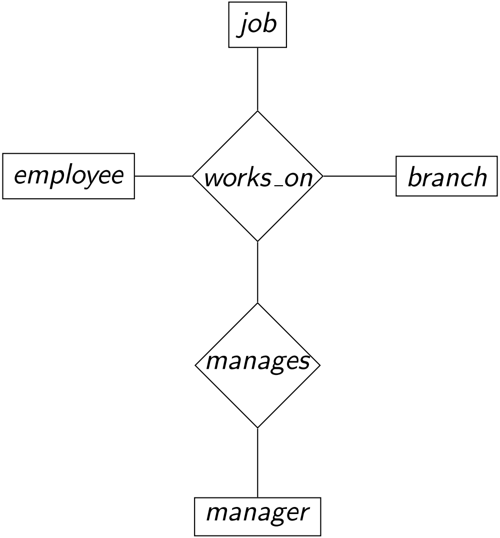
Consider the ternary relationship *works_on* on page 27. Suppose we want to record managers for tasks performed by an employee at a branch.



Aggregation

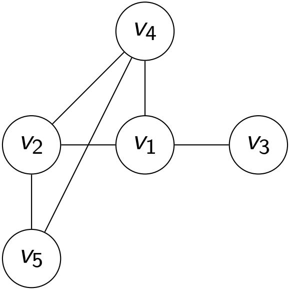
- Relationship sets *works_on* and manages represent overlapping information.
    - Every *manages* relationship corresponds to a *works_on* relationship.
    - However, some *works_on* relationships may not correspond to any *manages* relationships.
      So we cant discard the *works_on* relationship.
- Eliminate this redundancy via *aggregation*.
    - Treat relationship as an abstract entity.
    - Allows relationships between relationships.
    - Abstraction of relationship into new entity.
- Without introducing redundancy, the following diagram represents
    - an employee works on a particular job at a particular branch and
    - an employee, branch, job combination may have an associated manager.

Aggregation



E-R Diagrams and Graphs

- A **graph** G is defined as (V, E), where $V = \{v_1, v_2, \cdots\}$ is a set of **nodes** and E is a set of **edges**. Each edge connects two nodes.
- For example, the following graph is expressed by $V = \{v_1, v_2, v_3, v_4, v_5\}$ and $E = \{(v_1, v_2), (v_1, v_3), (v_1, v_4), (v_2, v_3), (v_2, v_5), (v_4, v_5)\}$.



E-R Diagrams and Graphs

- A **path** in a graph is a sequence of nodes such that any two consecutive nodes are connected by one edge in the graph.For example, v_4, v_2, v_1, v_3 is a path.
- A **cycle** is a path which begins and ends at the same node.For example, v_2, v_5, v_4, v_2 is a cycle.
- A **simple path/cycle** does not go through a same edge more than once.
- A graph is **connected** if there is a path between any pair of two nodes. Otherwise, it is **disconnected**.
- A graph is **acyclic** if it does not contain any simple cycle.

E-R Diagrams and Graphs

- E-R diagrams can be viewed as graphs. Entity sets, relationships, and attributes are nodes; while the connections are edges.
- What does it mean if a E-R diagram is disconnected?
  If a pair of entity sets are connected by a path in an E-R diagram, the entity sets are related, though perhaps indirectly. A disconnected graph implies that there are pairs of entity sets that are unrelated to each other. If we split the graph into connected components, we have, in effect, a separate database corresponding to each connected component.

E-R Diagrams and Graphs

- What does it mean if a E-R diagram is acyclic?
  As indicated in the answer to the previous part, a path in the graph between a pair of entity sets indicates a (possibly indirect) relationship between the two entity sets. If there is a cycle in the graph then every pair of entity sets on the cycle are related to each other in at least two distinct ways. If the E-R diagram is acyclic then there is a unique path between every pair of entity sets and, thus, a unique relationship between every pair of entity sets.

E-R Design Decisions

We will discuss the following issues:

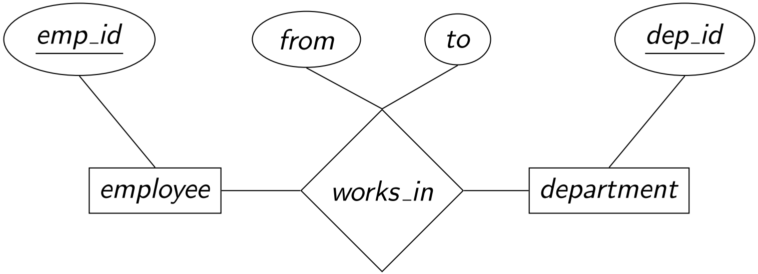
- an attribute or an entity for an object,
- an entity set or a relationship set for a real-world concept,
- ternary relationships versus binary relationships, and
- strong or weak entity sets.

Entity vs. Attribute

- Should *address* be an attribute of entity *employee* or an entity (connected to entity *employee* by relationship *live_at*)?
- Depends upon the use we want to make of address information, and the semantics of the data:
    - If we have several addresses per employee, *address* must be an entity OR a multi-valued attribute.
    - If the structure (city, street, etc.) is important, e.g., we want to retrieve employees in a given city, *address* must be modeled as an entity.

Entity vs. Attribute (Example)

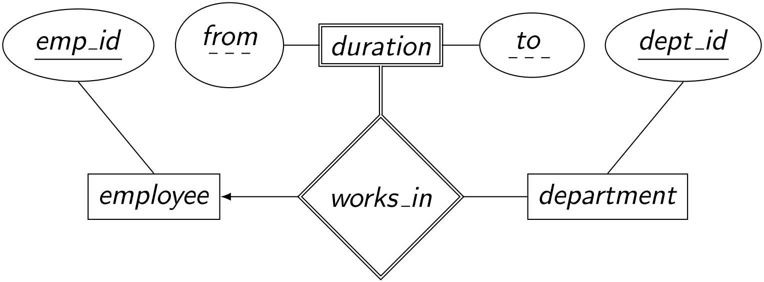
Suppose we want to design a database for employees working in departments. We are interested in the duration of each working period for every employee.



However, this model cannot show the case that an employee works in a department for multiple times non consecutively.

Entity vs. Attribute (Example)

Suppose we want to design a database for employees working in departments. We are interested in the duration of each working period for every employee.

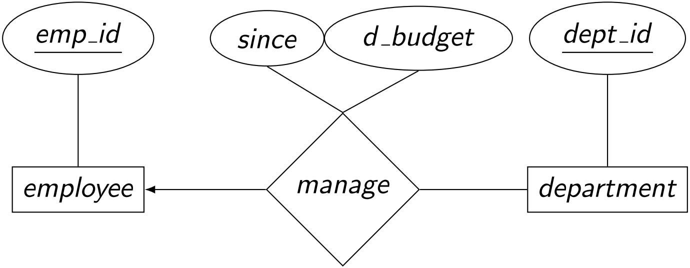


If we want to record multiple values of an attribute for each instance, we can introduce a new entity set.

Entity vs. Relationship

Suppose we want to design a database for a company with many departments. We want to show that some managers manage some departments.

The following ER diagram is okey if a manager gets a separate buget for each department.



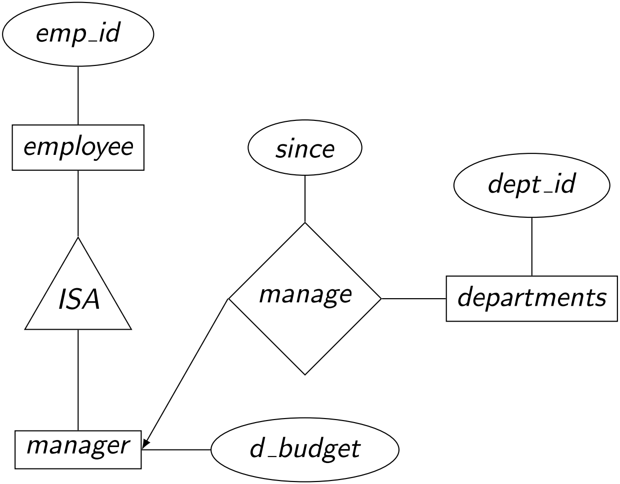
Entity vs. Relationship

What if the budget of each manager is the overall budget covering all managed departments? After convering the ER diagram to a relation model (the conversion procedure will be given by the following section), table *manage* is as follows.

emp_id	dept_id	since	d_budget
hg05uo	d_001	Jul. 2005	100,000
hg05uo	d_002	Jan. 2006	100,000
hg05uo	d_003	Sep. 2007	100,000

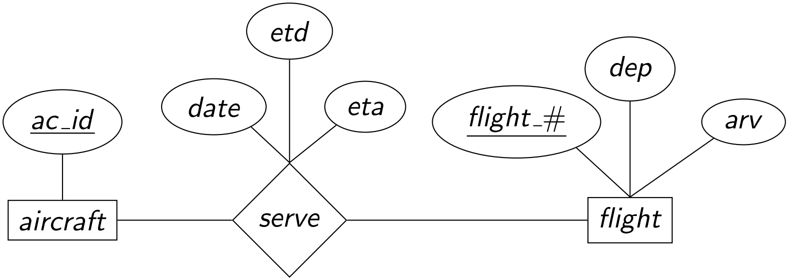
- **Redundancy**: The value of *d_budget* depends on *emp_id*. Same manager must have same budget.
- **Misleading**: The budget is the overall budget for all departments managed by one manager, not one single department.

A Better Design



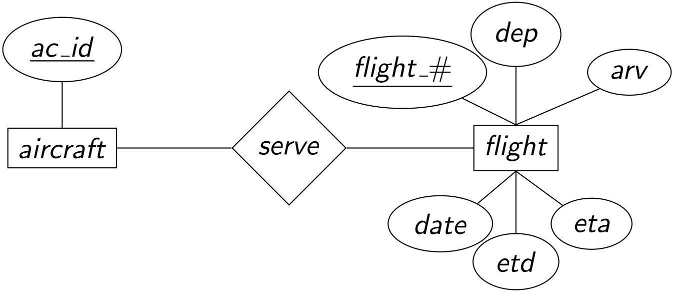
Binary vs. Ternary Relationships (Cont.)

Let's consider a database for aircrafts, flights, and the timetable. We want to record *aircraft id*, *flight number*, *departure*, *arrival*, *date*, *estimated time of departure(ETD)*, and *estimated time of arrival(ETA)*.



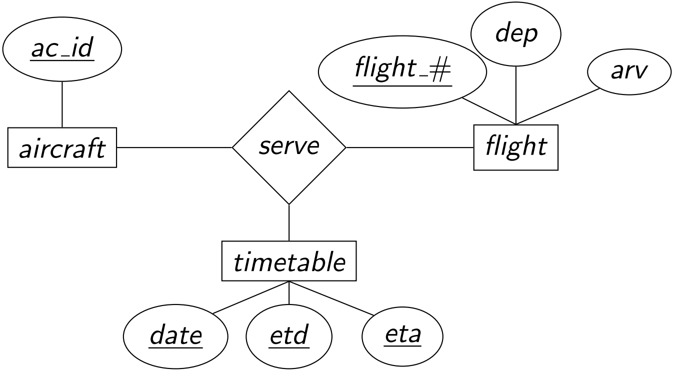
Problem: the date of each flight has to be different in the real case.

Binary vs. Ternary Relationships (Cont.)



Problem: *flight_#*, *dep*, *arv* will be heavily redundant.

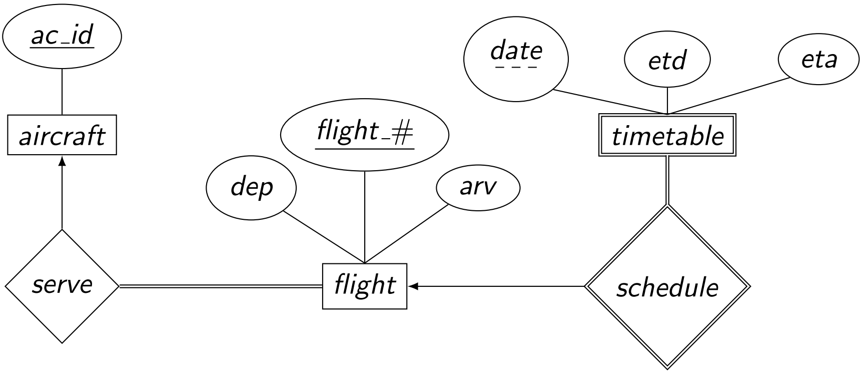
Binary vs. Ternary Relationships (Cont.)



Problem:

- *timetable* has to be a weak entity.
- *timetable* is in a total many-to-one relationship with *flight*.
- *flight* is in a total one-to-one relationship with *aircraft*.

Binary vs. Ternary Relationships (Cont.)

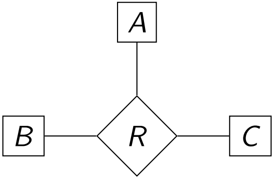


Binary vs. Ternary Relationships (Cont.)

- Using binary or ternary relationships depends on cases.
- Please consider the following examples and decide which relationship is the best and explain why.
  - Some *suppliers* sell manufacturing *parts* to *departments* in a company.
  - The relationship between *child* and *parents*.

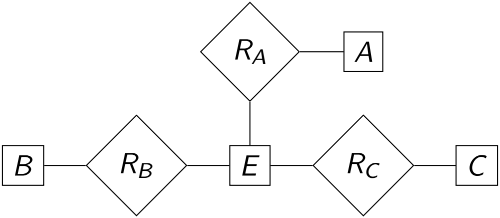
Converting Non-Binary Relationships to Binaries

- Any non-binary relationship can be converted to binary by introducing several artificial entity sets.
- For example a ternary relationship R among entity sets A, B, and C.



Converting Non-Binary Relationships to Binaries

- Create a new entity set E with an identity attribute.
- If the relationship set R has any attribute, assign them to E
- Create new relationship sets R_A between A and E, R_B between B and E, and R_C between C and E.
- For each relation (a_i, b_i, c_i) in the relationship set R, add
  - (e_i, a_i) in R_A,
  - (e_i, b_i) in R_B, and
  - (e_i, c_i) in R_C.



- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

Design Technique

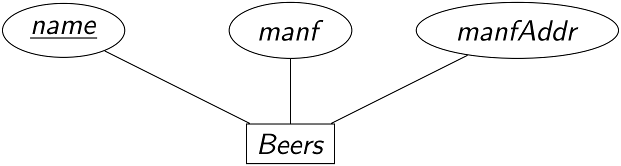
1. Express all constraints (you can express!)
2. Use and do not change terminology and class structure of the application domain.
3. Keep it simple.
   - Avoid defining entity types that do not serve any purpose.
   - Dont use an entity set when an attribute will do.
     Choose an entity set if it helps expressing constraints; otherwise, use an attribute.
4. Avoid redundancy (but derived attributes are okay)!
5. Limit the use of weak entity sets.

Avoiding Redundancy

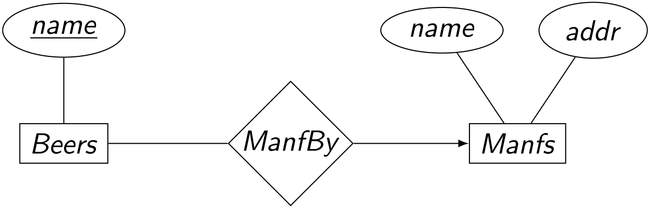
- Redundancy occurs when we say the same thing in two different ways.
- Redundancy wastes space and encourages inconsistency.
- The two instances of the same fact may become inconsistent if we change one and forget to change the other, related version.

Example

Is the following a good or bad design? And why?



Example



Don't overuse weak entity sets

- Beginning database designers often doubt that anything could be a key by itself.
  They make all entity sets weak, supported by all other entity sets to which they are linked.
- In reality, we usually create unique IDs for entity sets.
- We use weak entity sets when there is no capability of creating unique IDs.

Summary of ER Diagram Design

- Conceptual design follows requirements analysis.
  A design yields a high-level description of data to be stored.
- ER model is popular for conceptual design.
  Constructs are expressive and close to the way people think about their applications.
- Basic constructs: *entities*, *relationships*, and *attributes* (of entities and relationships).
- Some additional constructs: *weak entities*, *ISA hierarchies*, and *aggregation*.
- The ER diagram of the same problem can be variant.

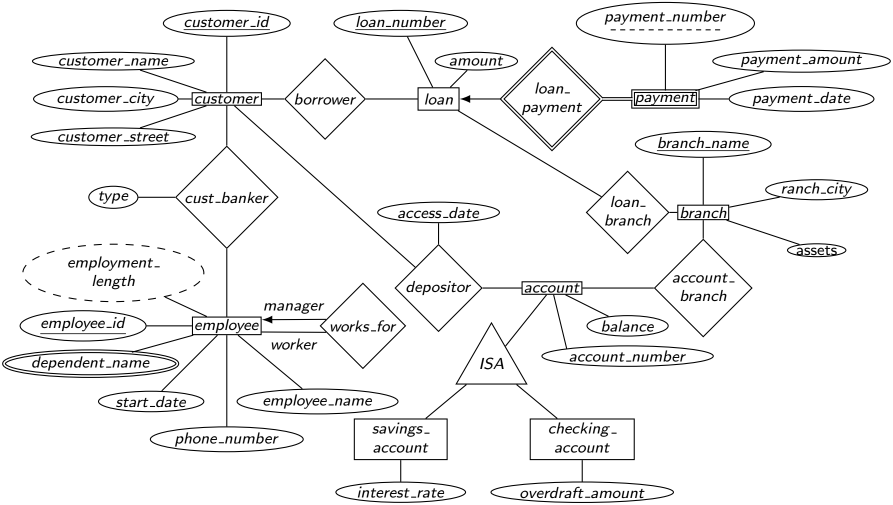
Summary of ER Diagram Design

- Several kinds of integrity constraints can be expressed in the ER model: *key constraints*, *participation constraints*, and *overlap/covering constraints* for ISA hierarchies.
- Some constraints (notably, *functional dependencies*) cannot be expressed in the ER model. (e.g., $z = x + y$)
- Constraints play an important role in determining the best database design for an enterprise.

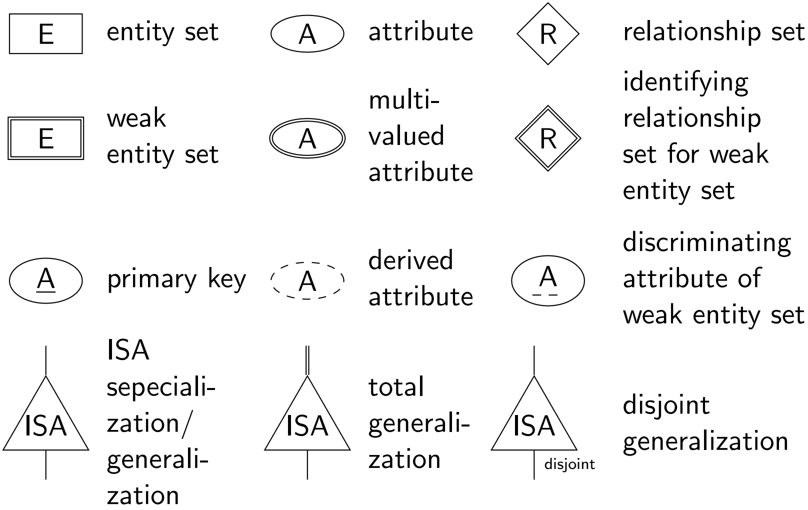
Summary of ER Diagram Design

- ER design is **subjective**. There are often many ways to model a given scenario! Analyzing alternatives can be tricky, especially for a large enterprise. Common choices include:
  - entity vs. attribute,
  - entity vs. relationship,
  - binary or n-ary relationship,
  - whether or not to use ISA hierarchies, and
  - whether or not to use aggregation.
- Ensuring good database design: resulting relational schema should be analyzed and refined further. FD information and normalization techniques are especially useful.

E-R Diagram for a Banking Enterprise

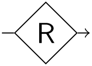


Summary of Symbols Used in E-R Diagram

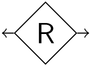


Summary of Symbols Used in E-R Diagram

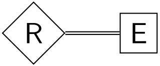
many_to_many relationship



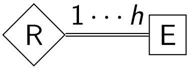
many_to_one relationship



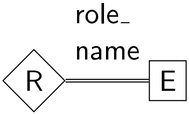
one_to_one relationship



total participation of entity set in relationship



cardinality limits

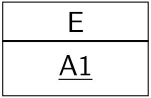


role indicator

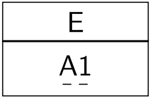
E-R Diagram by UML

- UML: a general-purpose modeling language in software engineering
- To visualize the design of a system
- Can be used for E-R diagram
- Entity sets, relationship sets, and mapping cardinality constraints are similar to E-R diagram.

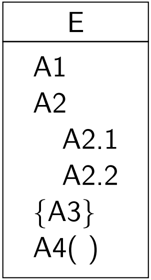
E-R Diagram by UML



E, the entity set
A1, the primary key

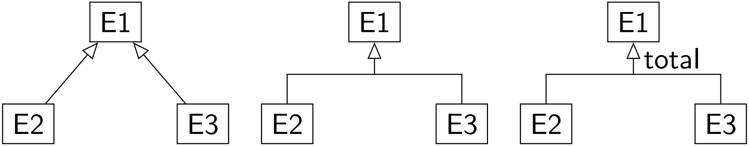


descriminating
attribute of
weak entity set



attributes:
simple (A1),
composite (A2),
multivalued (A3),
derived (A4)

E-R Diagram by UML



generalization
or specialization

disjoint
generalization

(total) disjoint
generalization

- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

Practice

- Consider a university database for the scheduling of classrooms for final exams. This database could be modeled as four entity sets
  - *exam* with attributes *exam_id* and *time*;
  - *course* with attributes *name*, *department* and *c_number*;
  - *section* with attributes *s_number* and *enrollment*;
  - *dependent* as a weak entity set on *course*; and
  - *room* with attributes *r_number*, *capacity*, and *building*.
- Show the E-R diagram illustrating the use of all three additional entity sets listed.

Practice

- Draw an E/R diagram to model project groups in CS3030. Keep in mind that
  - each enrolled student (identified by a PID) can work at most in one group;
  - each project, identified uniquely by its name, can have multiple groups working on it;
  - identify all the appropriate multiplicity and referential integrity constraints in the diagram; and
  - indicate key attributes in each entity set.

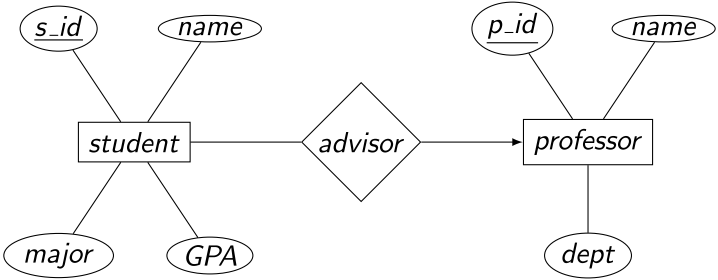
- Modeling
- Constraints
- E-R Diagram
- Weak Entity Sets
- Extended E-R Features
- Design Issues
- Design Technique
- Practice
- Reduction to Relations Schemas

E-R Diagram to Relation Model

Convert an E-R diagram into a relation model is not difficult. Basic ideas are as follows.

- Build a table for each entity set.
- Build a table for each relationship set if necessary (more on this later).
- Make a column in the table for each attribute in the entity set.
- Follow the indivisibility rule and ordering rule (more on this next chapter).
- Take care of primary keys.

Strong Entity Set



s_id	name	major	GPA
1234	John	CS	2.8
5678	Mary	EE	3.6

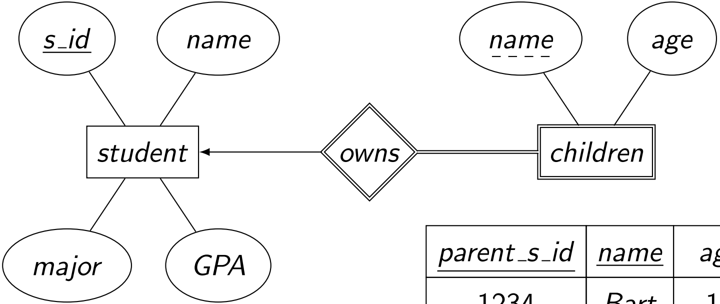
p_id	name	dept
999	Smith	Math
888	Lee	CS

Weak Entity Set

Weak entite set cannot exists alone. To build a table/schema for a weak entity set:

- construct a table with one column for each attribute in the weak entity set;
- including discriminator;
- augment one extra column on the right side of the table;
- put in there the primary key of the strong entity set that the weak entity set is depending on as a foreign key;
- then primary key of the weak entity set is the discriminator together with the foreign key.

Weat Entity Set



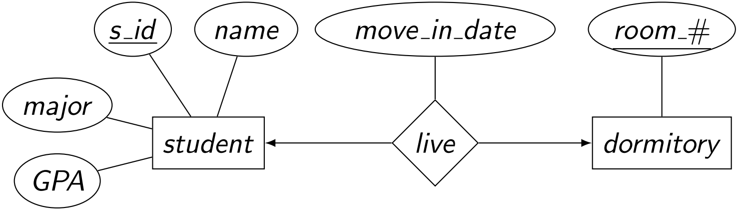
parent_s_id	name	age
1234	Bart	10
5678	Lisa	8

The primary key is (*parent_s_id*, *name*).

Relationship Set

- For one-to-one relationships with/out total participation,
    - build a table with two columns, one column for each participating entity sets primary key, and
    - add successive columns, one for each descriptive attributes of the relationship set (if any).
- For one-to-one relationships with one entity set having total participation,
    - augment one extra column on the right side of the table of the entity set with total participation, and
    - put in there the primary key of the entity set without complete participation as per to the relationship.

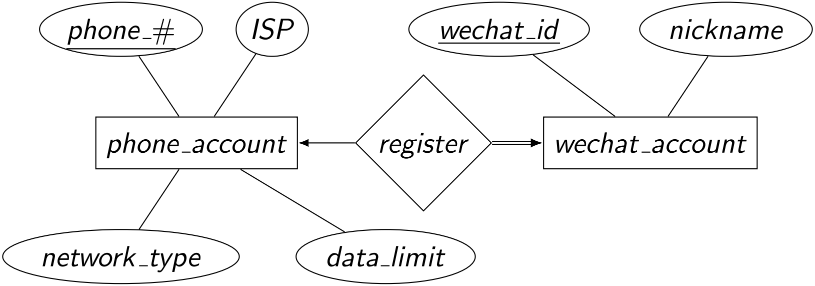
One-to-one Relationship Set



s_id	room_#	move_in_date
0430000123	502	10/09/2011
0312001015	409	07/09/2012

The primary key can either be (*s_id*) or (*room_#*).

One-to-one Relationship Set



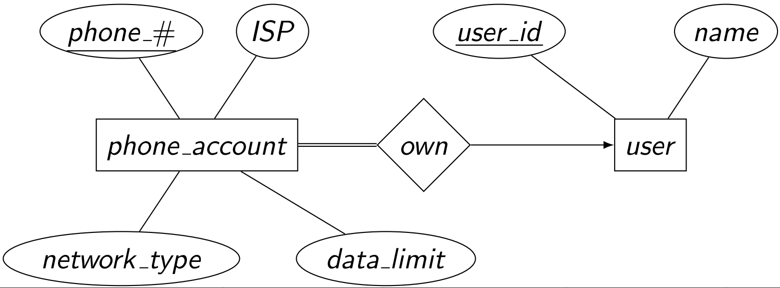
phone_#	*wechat_id*	*nickname*
13954925761	*wxid_32594*	*Dennis*
13522094687	*wxid_26518*	*Tomas*

The primary key is either (*phone_#*) or (*wechat_id*).

One-to-many Relationship Set

- For one-to-many relationships with/out total participation, it is the same thing as the conversion of one-to-one relationships.
- For those with one entity set having total participation on many side,
    - augment one extra column on the right side of the table of the entity set on the many side, and
    - put in there the primary key of the entity set on the one side as per to the relationship.

One-to-many Relationship Set



phone_#	ISP	network_type	data_limit	user_id
13954925761	Mobile	LTE	2	rh99hk
13522094687	Telecom	4G	1	rh99hk
13489526914	Unicom	CDMA	1.5	dz96js
15759210356	Mobile	5G	4	az97zj

The primary key has to be {*phone_#*}. {*user_id*} is the foreign key.

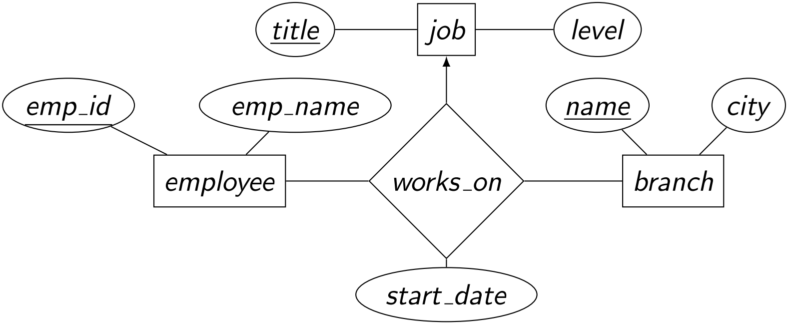
Many-to-many Relationship Set

- For many-to-many relationship, without total participation.
- The primary key of this new schema is the union of the foreign keys of both entity sets.

N-ary Relationship Set

- Build a new table with as many columns as there are attributes for the union of the primary keys of all participating entity sets.
- Create additional columns for descriptive attributes of the relationship set (if necessary).
- The primary key of this table is the union of all primary keys of entity sets that are on many side(s).

N-are Relationship Set



The primary key is $\{emp\_id, name\}$. And $\{title\}$ is a foreign key.

emp_id	*name*	*title*	*start_date*
sy93uk	*Victoria*	*director*	*May2005*
tr02mc	*Broadway*	*teller*	*Jun2005*
tr02mc	*Briand*	*manager*	*Feb2008*

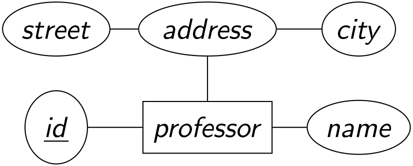
Weak Entity Set

You DONT have to build a table/schema for the identifying relationship set once you have built a table/schema for the corresponding weak entity set because

- it is a special case of one-to-many with total participation, and
- we want to minimize the redundancy.

Composite Attribute

- Relational model indivisibility rule applies
- One column for each component attribute
- NO column for the composite attribute itself



id	*name*	*city*	*street*
sy93uk	*Peters*	*Kingston*	*Monk*
kl00hz	*John*	*Hamilton*	*Yong*

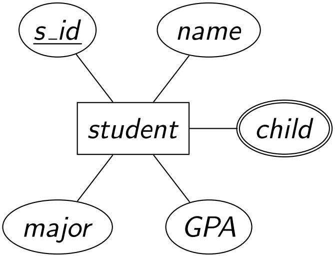
Multivalue Attribute

For each multivalue attribute in an entity set/relationship set,

- build a new relation schema with two columns;
- one column for the primary keys of the entity set/relationship set that has the multivalue attribute;
- another column for the multivalue attributes; and
- the primary key is the union of all attributes. Each cell of this column holds only one value. So each value is represented as an unique tuple

Since each cell of the column for the multivalue attribute holds only one value, each tuple is unique.

Multivalue attribute



s_id	child
043123	*Tommy*
043123	*Tonny*
031105	*Jimmy*
031105	*Lisa*

s_id	name	major	GPA
043123	*Jason*	*CS*	3.85
031105	*Thomas*	*EE*	3.12

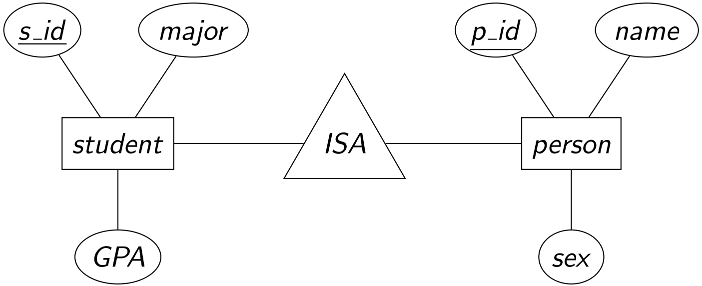
The primary key for the above table is $\{s\_id, child\}$.

Class Hierarchy

Two general approaches dependon disjointness and completeness.
For non-disjoint and/or non-complete class hierarchy,

- create a table for each super class entity set according to normal entity set translation method;
- create a table for each subclass entity set with a column for each of the attributes of that entity set;
- in the tables for subclass entity sets, create one extra column for each attributes of the primary key of the super class entity set; and
- the primary key from super class entity set is also used as the primary key for subclass entity sets.

Disjoint Class Hierarchy



p_id	s_id	major	GPA
99081263	0430010	CS	2.40
98122528	0312105	EE	3.12

p_id	name	sex
99081263	Beth	F
98122528	Tom	M

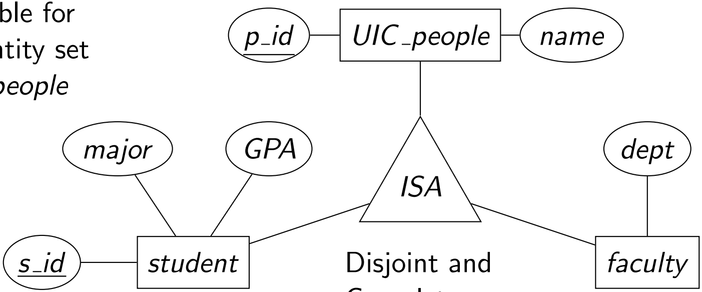
Class Hierarchy

For disjoint **AND** complete mapping class hierarchy,

- we do not create a table for the super class entity set,
- but create a table for each subclass entity set include all attributes of that subclass entity set and attributes of the superclass entity set.

Disjoint and Complete Class Hierarchy

No table for the entity set *UIC_people*



Disjoint and Complete

<u>*p_id*</u>	*name*	*s_id*	*major*	*GPA*
99081263	*Beth*	0430010	*CS*	2.40
98122528	*Tom*	0312105	*EE*	3.12

<u>*p_id*</u>	*name*	*dept*
74093058	*Brock*	*CS*
69032730	*Issac*	*EE*

Aggregation

Same as *n*-ary relationship set

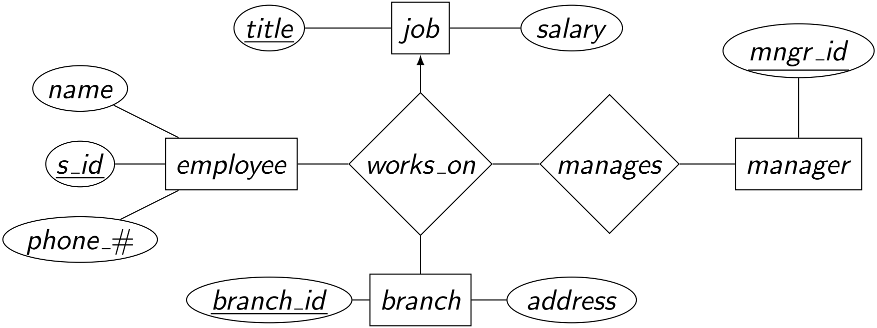


Table *manages*:

p_id	*branch_id*	*mngr_id*
84021943	045	691030
88060927	045	691030

End of the Chapter