Dept. of Computer Science and Engineering
University of Rajshahi
www.ru.ac.bd

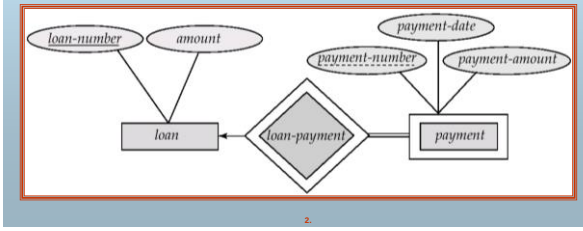
Dr. Shamim Ahmad

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***
  - it must relate to the identifying entity set via a total, one-to-many relationship set from the identifying to the weak entity set
  - **Identifying relationship** depicted using 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 set's discriminator.

Weak Entity Sets (Cont.)

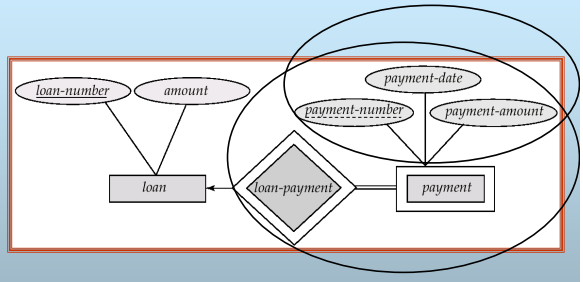
- We depict a weak entity set by double rectangles.
- We underline the discriminator of a weak entity set with a dashed line.
- ***payment-number*** **– discriminator** of the *payment* entity set
- Primary key for *payment* – (***loan-number, payment-number***)



Weak Entity Sets (Cont.)

- multivalued,composite attribute
- A weak entity set may be more appropriately modeled as an attribute if it participates in only the **identifying relationship**, and if it has few attributes.

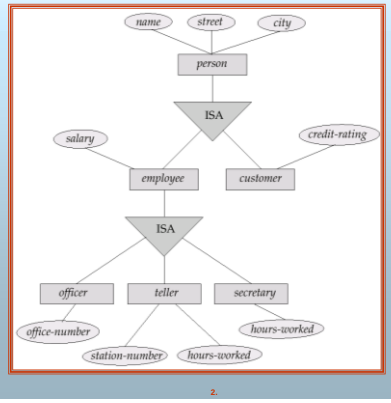
Weak Entity Sets (Cont.)



Extended E-R Features

- **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*).
  - **superclass-subclass relationship.**
  - 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



Extended E-R Features (Cont.)

- **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.

Extended E-R Features (Cont.)

- Attribute Inheritance
  - The attributes of the higher-level entity sets are said to be inherited by the lower-level entity sets.
  - A lower-level entity set(or subclass)also inherits participation in the relationship sets in which its higher-level entity(or superclass) participation.
  - Inheritance
    - A higher-level entity set with attributes and relationships that apply to all of its lower-level entity sets
    - Lower-level entity sets with distinctive features that apply only within a particular lower-level entity set
  - Single inheritance
  - Multiple inheritance

2.

Extended E-R Features (Cont.)

- Constraints on Generalization
  - Constraint on which entities can be members of a given lower-level entity set.
    - condition-defined: all the lower-level entities are evaluated on the basis of the same attribute, this type of generalization is said to be attribute-defined.
      - E.g. all customers **over 65** years are members of *senior-citizen* entity set; *senior-citizen* ISA *person*.
    - user-defined
  - Constraint on whether or not entities may belong to more than one **lower-level entity** set within a single generalization. (Example: Work Team After 3 Months)

2.

Extended E-R Features (Cont.)

- 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

2.

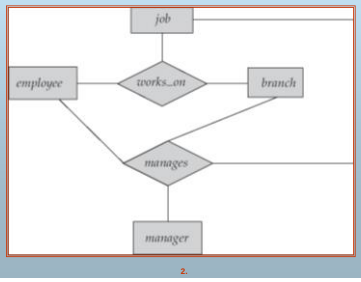
Extended E-R Features (Cont.)

- Constraints on Generalization
  - Disjoint
    - an entity can belong 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 set (Manager in Every Work Team)

2.

Aggregation

- Consider the ternary relationship *works_on*, which we saw earlier
- Suppose we want to record managers for tasks performed by an employee at a branch



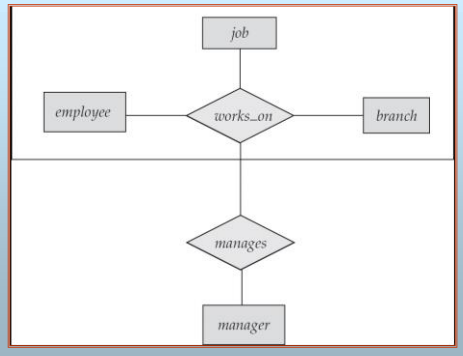
2.

Aggregation (Cont.)

- 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 can't 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
  - An employee, branch, job combination may have an associated manager

2.

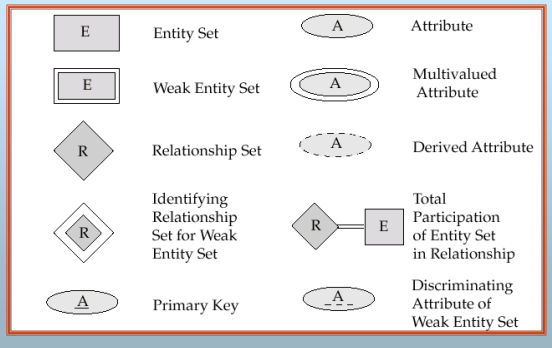
E-R Diagram With Aggregation



2.

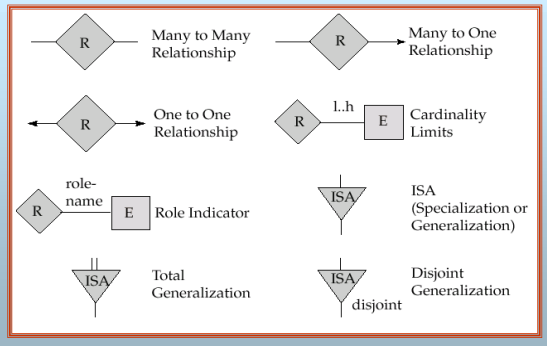
Extended E-R Features (Cont.)

- **Alternative E-R Notations**

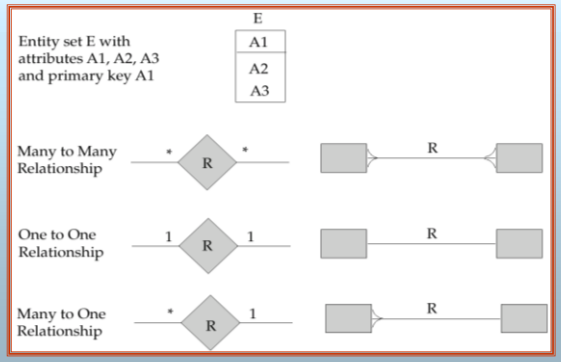


2.

Extended E-R Features (Cont.)



Extended E-R Features (Cont.)



Design of an E-R Database Schema

Among the designer's decisions are:

- The use of an attribute or entity set to represent an object.
- Whether a real-world concept is best expressed by an entity set or a relationship set.
- The use of a ternary relationship versus a pair of binary relationships.
- The use of a strong or weak entity set.
- The use of specialization/generalization – contributes to modularity in the design.
- The use of aggregation – can treat the aggregate entity set as a single unit without concern for the details of its internal structure.

Design of an E-R Database Schema (Cont.)

- Design Phases
  - Conceptual-design(specification of functional requirements)
  - Logical-design phases
  - Physical-design phase
- Database design for Banking Enterprise
  - Data Requirements
    - The bank is organized into branches.
    - Bank customers are identified by their *customer-id* values.
    - Bank employees are identified by their *employee-id* values.

Design of an E-R Database Schema (Cont.)

- The bank offers two types of *accounts-saving* and *checking accounts*.
- A loan originates at a particular branch and can be held by one or more customers.

Entity Sets Designation

- The *branch* entity set.
- The *customer* entity set.
- The *employee* entity set.
- Two *account* entity sets.
- The *loan* entity set.
- The weak entity set *loan-payment*.

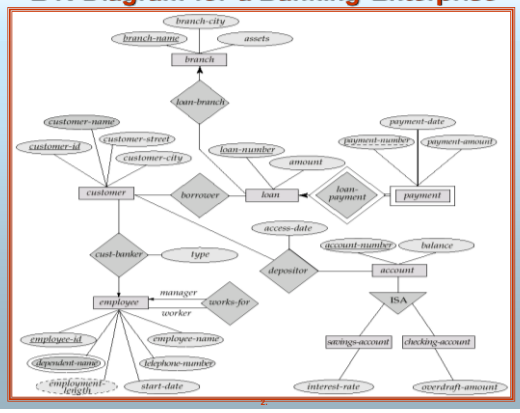
Design of an E-R Database Schema (Cont.)

Relationship Sets Designation

- *borrower*.
- *loan-branch*.
- *loan-payment*.
- *depositor*.
- *cust-banker*.
- *works-for*.

E-R Diagram

E-R Diagram for a Banking Enterprise



Reduction of an E-R Schema to Tables

- **Primary keys** allow entity sets and relationship sets to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- **Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram**.

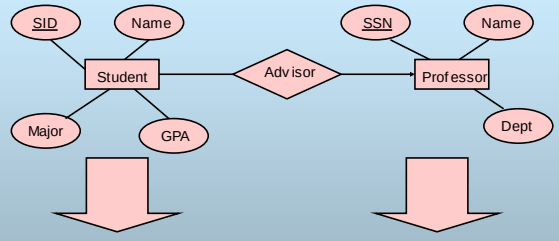
Representing strong Entity Sets as Tables

- **A strong entity** set reduces to a table with the same attributes.

customer-id	customer-name	customer-street	customer-city
019-28-3746	Smith	North	Rye
182-73-6091	Turner	Putnam	Stamford
192-83-7465	Johnson	Alma	Palo Alto
244-66-8800	Curry	North	Rye
321-12-3123	Jones	Main	Harrison
335-57-7991	Adams	Spring	Pittsfield
336-66-9999	Lindsay	Park	Pittsfield
677-89-9011	Hayes	Main	Harrison
963-96-3963	Williams	Nassau	Princeton

Representing strong Entity Sets as Tables

- **A strong entity** set reduces to a table with the same attributes.



SID	Name	Major	GPA
1234	John	CS	2.8
5678	Mary	EE	3.6

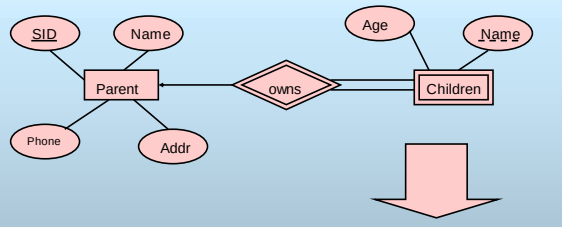
SSN	Name	Dept
9999	Smith	Math
8888	Lee	CS

Representing Weak Entity Sets

- **A weak entity** set becomes a table that **includes a column for the primary key** of the **identifying strong entity set**

loan-number	payment-number	payment-date	payment-amount
L-11	53	7 June 2001	125
L-14	69	28 May 2001	500
L-15	22	23 May 2001	300
L-16	58	18 June 2001	135
L-17	5	10 May 2001	50
L-17	6	7 June 2001	50
L-17	7	17 June 2001	100
L-23	11	17 May 2001	75
L-93	103	3 June 2001	900
L-93	104	13 June 2001	200

Representing Weak Entity Sets



Age	Name	Parent_SID
10	Bart	1234
8	Lisa	5678

*** Primary key of *Children* is *Parent_SID + Name***

Representing Relationship Sets as Tables

- A **many-to-many relationship** set is represented as a table with columns for the **primary keys of the two participating entity sets**, and any descriptive attributes of the relationship set.
- E.g.: table for relationship set *borrower*

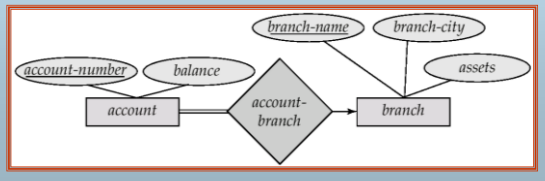
customer-id	*loan-number*
019-28-3746	L-11
019-28-3746	L-23
244-66-8800	L-93
321-12-3123	L-17
335-57-7991	L-16
555-55-5555	L-14
677-89-9011	L-15
963-96-3963	L-17

Many-to-many Relationship Sets

- For many-to-many relationship
  - Same thing as one-to-one relationship without total participation.
  - Primary key of this new schema is the union of the foreign keys of both entity sets.

Redundancy of Tables

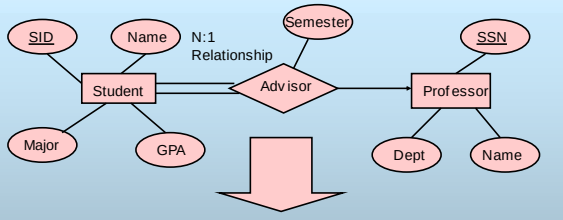
- **Many-to-one** and **one-to-many** relationship sets that are total on the **many-side** can be represented by adding an **extra attribute** to the many side, containing the **primary key of the one side**
- E.g.: Instead of creating a table for relationship *account-branch*, add an attribute *branch* to the entity set *account*



Many-to-one and One-to-many

- **For one-to-many** relationship w/out total participation
  - Same thing as one-to-one
- For **one-to-many/many-to-one** relationship 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, put in there the primary key of the entity set on the "one" side as per to the relationship.

Example – Many-to-One Relationship Set



SID	Name	Major	GPA	Pro_SSN	Ad_Sem
9999	Bart	Economy	-4.0	123-456	Fall 2006
8888	Lisa	Physics	4.0	567-890	Fall 2005

* Primary key of this table is *SID*

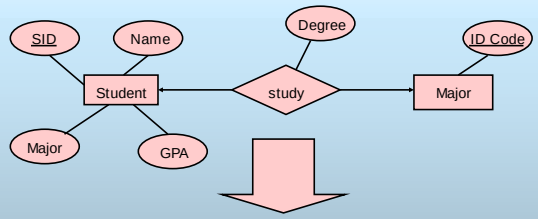
Redundancy of Tables (Cont.)

- For **one-to-one** relationship sets, **either side can be chosen to act as the "many" side**
  - That is, **extra attribute can** be added to **either of the tables corresponding to the two entity sets**

One-to-one Relationship Set

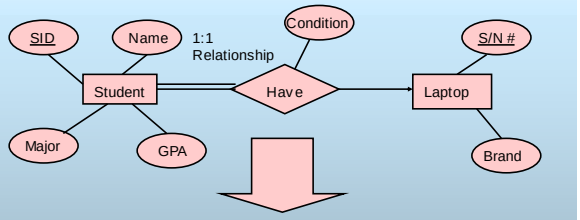
- For one-to-one relationship w/out total participation
  - Build a table with two columns, one column for each participating entity set's primary key. Add successive columns, one for each descriptive attributes of the relationship set (if any).
- For one-to-one relationship with one entity set having total participation
  - Augment one extra column on the right side of the table of the entity set with total participation, put in there the primary key of the entity set without complete participation as per to the relationship

Example One-to-One Relationship Set



SID	Maj_ID_Co	S_Degree
9999	07	1234
8888	05	5678

Example One-to-One Relationship Set



SID	Name	Major	GPA	LP_S/N	Hav_Cond
9999	Bart	Economy	-4.0	123-456	Own
8888	Lisa	Physics	4.0	567-890	Loan

* Primary key can be either *SID* or *LP_S/N*

2.

Composite and Multivalued Attributes

- **Composite** attributes are **flattened** out by creating a separate attribute for **each component attribute**
  - E.g. given entity set *customer* with composite attribute *name* with component attributes *first-name* and *last-name* the table corresponding to the entity set has two attributes
    name.first-name and *name.last-name*

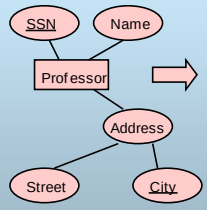
2.

Composite and Multivalued Attributes

- A **multivalued** attribute M of an entity E is represented by a **separate table EM**
  - Table EM has attributes corresponding to the **primary key of E** and an attribute corresponding to **multivalued attribute M**
  - Each value of the **multivalued attribute** maps to a **separate row** of the table EM

2.

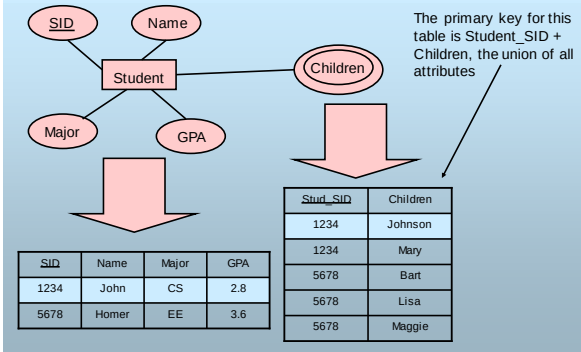
Representing Composite Attribute



SSN	Name	Street	City
9999	Dr. Smith	50 1st St.	Fake City
8888	Dr. Lee	1 B St.	San Jose

2.

Representing Multivalue Attribute



SID	Name	Major	GPA
1234	John	CS	2.8
5678	Homer	EE	3.6

Stud_SID	Children
1234	Johnson
1234	Mary
5678	Bart
5678	Lisa
5678	Maggie

2.

Representing Specialization as Tables

- Method 1:
  - **Form a table for the higher level entity set**
  - **Form a table for each lower level entity set, include primary key of higher level entity set and local attributes**

table	table attributes
person	*name, street, city*
customer	*name, credit-rating*
employee	*name, salary*

  - **Drawback: getting information about, e.g., *employee* requires accessing two tables**

2.

Representing Specialization as Tables (Cont.)

- Method 2:
  - Form a table for each entity set with **all local and inherited attributes**

table	table attributes
person	*name, street, city*
customer	*name, street, city, credit-rating*
employee	*name, street, city, salary*

  - If specialization is total, table for generalized entity (*person*) not required to store information
    - Can be defined as a "view" relation containing union of specialization tables
  - Drawback: street and city may be stored redundantly for *persons* who are both *customers* and *employees*

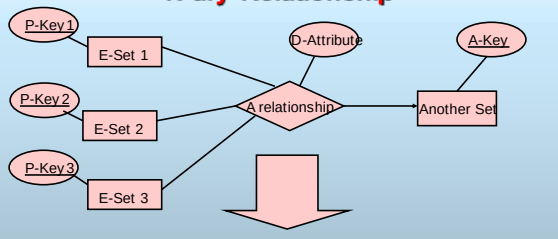
2.

Representing Relationship Set N-ary Relationship

- Intuitively Simple
  - Build a new table with as many columns as there are attributes for the union of the primary keys of all participating entity sets.
  - Augment 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
  - That is it, we are done.

2.

Representing Relationship Set N-ary Relationship



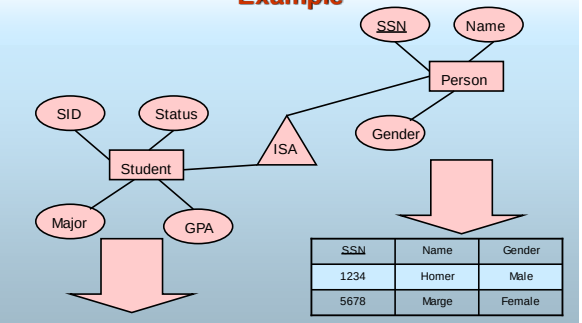
P-Key1	P-Key2	P-Key3	A-Key	D-Attribute
9999	8888	7777	6666	Yes
1234	5678	9012	3456	No

* Primary key of this table is *P-Key1 + P-Key2 + P-Key3*

Representing Class Hierarchy

- Two general approaches depending on disjointness and completeness
  - For disjoint **AND** complete mapping class hierarchy:
  - DO NOT create a table for the super class entity set
  - Create a table for each subclass entity set include all attributes of that subclass entity set and attributes of the superclass entity set

Example



SSN	Name	Gender
1234	Homer	Male
5678	Marge	Female

SSN	SID	Status	Major	GPA
1234	9999	Full	CS	2.8
5678	8888	Part	EE	3.6

- Fourth Edition
- Problem #
  - 2.12
  - 2.14
  - 2.26