CS425 – Fall 2013
Boris Glavic
Chapter 7: Entity-Relationship Model

Partially taken from
Klaus R. Dittrich

modified from:
Database System Concepts, 6th Ed.
©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use
Chapter 7: Entity-Relationship Model

- Design Process
- Modeling
- Constraints
- E-R Diagram
- Design Issues
- Weak Entity Sets
- Extended E-R Features
- Design of the Bank Database
- Reduction to Relation Schemas
- Database Design
- UML
Database Design

World

???

Relational DB schema
First: need to develop a “mind”-model based on a requirement analysis.
Requirement Analysis Example
Zoo

- The zoo stores information about animals, cages, and zoo keepers.
- Animals are of a certain species and have a name. For each animal we want to record its weight and age.
- Each cage is located in a section of the zoo. Cages can house animals, but there may be cages that are currently empty. Cages have a size in square meter.
- Zoo keepers are identified by their social security number. We store a first name, last name, and for each zoo keeper. Zoo keepers are assigned to cages they have to take care of (clean, ...). Each cage that is not empty has a zoo keeper assigned to it. A zoo keeper can take care of several cages. Each zoo keeper takes care of at least one cage.
Let’s do it!
Second: Formalize this model by developing a conceptual model.
Second: Formalize this model by developing a conceptual model.

World

Requirement Analysis

“Mind” Model

Conceptual modeling

Conceptual Model

Logical modeling (possibly automated)

Relational DB schema

English (e.g.)

ER model

SQL (e.g.)
Modeling – ER model

- A database can be modeled as:
  - a collection of entities,
  - relationship among entities.
- An entity is an object that exists and is distinguishable from other objects.
  - Example: specific person, company, event, plant
- Entities have attributes
  - Example: people have names and addresses
- 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 *instructor and student*

**instructor**

<table>
<thead>
<tr>
<th>instructor_ID</th>
<th>instructor_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
</tr>
</tbody>
</table>

**student**

<table>
<thead>
<tr>
<th>student-ID</th>
<th>student_name</th>
</tr>
</thead>
<tbody>
<tr>
<td>98988</td>
<td>Tanaka</td>
</tr>
<tr>
<td>12345</td>
<td>Shankar</td>
</tr>
<tr>
<td>00128</td>
<td>Zhang</td>
</tr>
<tr>
<td>76543</td>
<td>Brown</td>
</tr>
<tr>
<td>76653</td>
<td>Aoi</td>
</tr>
<tr>
<td>23121</td>
<td>Chavez</td>
</tr>
<tr>
<td>44553</td>
<td>Peltier</td>
</tr>
</tbody>
</table>
Relationship Sets

A **relationship** is an association among several entities.

Example:

- 44553 (Peltier) **advisor** 22222 (Einstein)
- student entity  relationship set  instructor entity

A **relationship set** is a mathematical relation among \( n \geq 2 \) entities, each taken from entity sets

\[
\{(e_1, e_2, \ldots, e_n) \mid e_1 \in E_1, e_2 \in E_2, \ldots, e_n \in E_n\}
\]

where \((e_1, e_2, \ldots, e_n)\) is a relationship

- Example:

  \[(44553, 22222) \in advisor\]
### Relationship Set advisor

<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766 Crick</td>
<td>98988 Tanaka</td>
</tr>
<tr>
<td>45565 Katz</td>
<td>12345 Shankar</td>
</tr>
<tr>
<td>10101 Srinivasan</td>
<td>00128 Zhang</td>
</tr>
<tr>
<td>98345 Kim</td>
<td>76543 Brown</td>
</tr>
<tr>
<td>76543 Singh</td>
<td>76653 Aoi</td>
</tr>
<tr>
<td>22222 Einstein</td>
<td>23121 Chavez</td>
</tr>
<tr>
<td></td>
<td>44553 Peltier</td>
</tr>
</tbody>
</table>
An attribute can also be property of a relationship set.

For instance, the advisor relationship set between entity sets instructor and student may have the attribute date which tracks when the student started being associated with the advisor.
Degree of a Relationship Set

- **binary relationship**
  - involve two entity sets (or degree two).
- Relationships between more than two entity sets are rare. Most relationships are binary. (More on this later.)
  - Example: *students* work on research *projects* under the guidance of an *instructor*.
  - relationship *proj_guide* is a ternary relationship between *instructor, student, and project*
Attributes

- An entity is represented by a set of attributes, that are descriptive properties possessed by all members of an entity set.
  - Example:
    
    \[
    \text{instructor} = (\text{ID}, \text{name}, \text{street}, \text{city}, \text{salary}) \\
    \text{course} = (\text{course\_id}, \text{title}, \text{credits})
    \]

- **Domain** – the set of permitted values for each attribute

- Attribute types:
  - **Simple** and **composite** attributes.
  - **Single-valued** and **multivalued** attributes
    - Example: multivalued attribute: \textit{phone\_numbers}
  - **Derived** attributes
    - Can be computed from other attributes
    - Example: \textit{age}, given \textit{date\_of\_birth}
Composite Attributes

Composite attributes

name

first_name  middle_initial  last_name

address

street  city  state  postal_code

street_number  street_name  apartment_number

Component attributes
Mapping Cardinality Constraints

- Express the number of entities to which another entity can be associated via a relationship set.

- For a binary relationship set the mapping cardinality must be one of the following types:
  - One to one (1-1)
  - One to many (1-N)
  - Many to one (N-1)
  - Many to many (N-M)
Mapping Cardinalities

(a) One to one

(b) One to many

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

(a) One to one

(b) One to many

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

Many to one

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

Many to many
Mapping Cardinalities Example

Employee \rightarrow Department

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

Student \rightarrow Course

Many to many
### Mapping Cardinality Constraints Cont.

- **What if we allow some elements to not be mapped to another element?**
  - E.g., 0:1 → 1

- For a binary relationship set, the mapping cardinality must be one of the following types:

#### 1-1
- 1-1
- 0:1-1
- 1-0:1
- 0:1-0:1

#### 1-N
- 0:1-N
- 0:1-0:N
- 1-N
- 1-0:N

#### N-1
- N-1
- N-0:1
- 0:N-1
- 0:N-0:1

#### N-M
- N-M
- N-0:M
- 0:N-M
- 0:N-0:M
Mapping Cardinality Constraints Cont.

- Typical Notation
  - (0:1) – (1:N)
A **super key** of an entity set is a set of one or more attributes whose values uniquely determine each entity.

A **candidate key** of an entity set is a minimal super key

- *ID* is candidate key of *instructor*
- *course_id* is candidate key of *course*

Although several candidate keys may exist, one of the candidate keys is selected to be the **primary key**.

**Note:** Basically the same as for relational model
Keys for Relationship Sets

- The combination of primary keys of the participating entity sets forms a super key of a relationship set.
  - \((s\_id, i\_id)\) is the super key of \textit{advisor}
  - \textit{NOTE: this means a pair of entities can have at most one relationship in a particular relationship set.}
    - Example: if we wish to track multiple meeting dates between a student and her advisor, we cannot assume a relationship for each meeting. We can use a multivalued attribute though or model meeting as a separate entity

- Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys

- Need to consider semantics of relationship set in selecting the \textit{primary key} in case of more than one candidate key
Must consider the mapping cardinality of the relationship set when deciding what are the candidate keys

- 1-1: both primary keys are candidate keys
  - Example: `hasBc`: (Person-Birthcertificate)

- N-1: the N side is the candidate key
  - Example: `worksFor`: (Instructor-Department)

- N-M: the combination of both primary keys
  - Example: `takes`: (Student-Course)
Redundant Attributes

- Suppose we have entity sets
  - *instructor*, with attributes including *dept_name*
  - *department*

  and a relationship
  - *inst_dept* relating *instructor* and *department*

- Attribute *dept_name* in entity *instructor* is redundant since there is an explicit relationship *inst_dept* which relates instructors to departments
  - The attribute replicates information present in the relationship, and should be removed from *instructor*
  - BUT: when converting back to tables, in some cases the attribute gets reintroduced, as we will see.
E-R Diagrams

- Rectangles represent entity sets.
- Diamonds represent relationship sets.
- Attributes listed inside entity rectangle.
- Underline indicates primary key attributes.
### Entity With Composite, Multivalued, and Derived Attributes

<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ID</strong></td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
<tr>
<td>middle_initial</td>
</tr>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
</tr>
<tr>
<td>apt_number</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state</td>
</tr>
<tr>
<td>zip</td>
</tr>
<tr>
<td>{ phone_number }</td>
</tr>
<tr>
<td>date_of_birth</td>
</tr>
<tr>
<td>age ( )</td>
</tr>
</tbody>
</table>
Entity With Composite, Multivalued, and Derived Attributes

<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
<tr>
<td>middle_initial</td>
</tr>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
</tr>
<tr>
<td>apt_number</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state</td>
</tr>
<tr>
<td>zip</td>
</tr>
<tr>
<td>{ phone_number }</td>
</tr>
<tr>
<td>date_of_birth</td>
</tr>
<tr>
<td>age ( )</td>
</tr>
</tbody>
</table>

Composite:
- ID
- name
- first_name
- middle_initial
- last_name
- address
- street
- street_number
- street_name
- apt_number
- city
- state
- zip

Derived:
- { phone_number }
- date_of_birth
- age ( )

Multi-valued:
- street

Multi-valued:
- street_number

Composite:
- { phone_number }
Relationship Sets with Attributes

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>
```

 отношения adviser  

date

ID
name
tot_cred
Roles

- Entity sets of a relationship need not be distinct
  - Each occurrence of an entity set plays a “role” in the relationship
- The labels “course_id” and “prereq_id” are called roles.
Cardinality Constraints

- We express cardinality constraints by drawing either a directed line (→), signifying “one,” or an undirected line (—), signifying “many,” between the relationship set and the entity set.

- One-to-one relationship:
  - A student is associated with at most one instructor via the relationship advisor
  - A student is associated with at most one department via stud_dept
One-to-One Relationship

- one-to-one relationship between an instructor and a student
  - an instructor is associated with at most one student via advisor
  - and a student is associated with at most one instructor via advisor

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>
```

Advisor relationship:

- Instructor is associated with at most one student via an advisor
- Student is associated with at most one instructor via an advisor
One-to-Many Relationship

- one-to-many relationship between an instructor and a student
  - an instructor is associated with several (including 0) students via advisor
  - a student is associated with at most one instructor via advisor,
In a many-to-one relationship between an instructor and a student,

- an instructor is associated with at most one student via advisor,
- and a student is associated with several (including 0) instructors via advisor.

```
instructor
ID
name
salary

advisor

student
ID
name
tot_cred
```
Many-to-Many Relationship

- An instructor is associated with several (possibly 0) students via advisor
- A student is associated with several (possibly 0) instructors via advisor

<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>

advisor
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
  ● E.g., participation of section in sec_course is total
    ‣ every section must have an associated course

■ Partial participation: some entities may not participate in any relationship in the relationship set
  ● Example: participation of instructor in advisor is partial

```
course
<table>
<thead>
<tr>
<th>course_id</th>
<th>title</th>
<th>credits</th>
</tr>
</thead>
</table>

section
<table>
<thead>
<tr>
<th>sec_id</th>
<th>semester</th>
<th>year</th>
</tr>
</thead>
</table>

sec_course
```

""
Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>advisor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>0..*</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td></td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>1..1</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>
```


Alternative Notation for Cardinality Limits

- Alternative Notation

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>

0..*          1..1
(0,n)          (1,1)

advisor

```
We allow at most one arrow out of a ternary (or greater degree) relationship to indicate a cardinality constraint.

E.g., an arrow from `proj_guide` to `instructor` indicates each student has at most one guide for a project.

If there is more than one arrow, there are two ways of defining the meaning.

- E.g., a ternary relationship `R` between `A`, `B` and `C` with arrows to `B` and `C` could mean:
  1. each `A` entity is associated with a unique entity from `B` and `C` or
  2. each pair of entities from `(A, B)` is associated with a unique `C` entity, and each pair `(A, C)` is associated with a unique `B` entity.

- Each alternative has been used in different formalisms.

- To avoid confusion we outlaw more than one arrow.

Better to use cardinality constraints such as `(0,n)`.
Let’s design an ER-model for parts of the university database

Partially taken from
Klaus R. Dittrich

modified from:
Database System Concepts, 6th Ed.
©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use
Let's design an ER-model for parts of the university database.

1) Identify Entities
2) Identify Relationship
3) Determine Attributes
4) Determine Cardinality Constraints

Partially taken from Klaus R. Dittrich

modified from: Database System Concepts, 6th Ed.

©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use
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 that are associated with the same entity of the identifying 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 underline the discriminator of a weak entity set with a dashed line.
- We put the identifying relationship of a weak entity in a double diamond.
- Primary key for section – (course_id, sec_id, semester, year)
Note: the primary key of the strong entity set is not explicitly stored with the weak entity set, since it is implicit in the identifying relationship.

If course_id were explicitly stored, section could be made a strong entity, but then the relationship between section and course would be duplicated by an implicit relationship defined by the attribute course_id common to course and section.
Reduction to Relational Schemas
Reduction to Relation Schemas

- Entity sets and relationship sets can be expressed uniformly as *relation schemas* that represent the contents of the database.

- A database which conforms to an E-R diagram can be represented by a collection of relation schemas.

- For each entity set and relationship set there is a unique relation schema that is assigned the name of the corresponding entity set or relationship set.
A strong entity set reduces to a schema with the same attributes
student(ID, name, tot_cred)

A weak entity set becomes a table that includes a column for the primary
key of the identifying strong entity set
section (course_id, sec_id, sem, year)
A many-to-many relationship set is represented as a schema with attributes for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.

Example: schema for relationship set "advisor"

\[ \text{advisor} = (s\_id, i\_id) \]
Redundancy of Schemas

- 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.

- Example: Instead of creating a schema for relationship set inst_dept, add an attribute dept_name to the schema arising from entity set instructor.
Redundancy of Schemas (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
  - If the relationship is total in both sides, the relation schemas from the two sides can be merged into one schema
- If participation is *partial* on the “many” side, replacing a schema by an extra attribute in the schema corresponding to the “many” side could result in null values
- The schema corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - Example: The *section* schema already contains the attributes that would appear in the *sec_course* schema
Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute.
  - Example: given entity set `instructor` with composite attribute `name` with component attributes `first_name` and `last_name` the schema corresponding to the entity set has two attributes `name_first_name` and `name_last_name`.
    - Prefix omitted if there is no ambiguity.

- Ignoring multivalued attributes, extended instructor schema is:
  - `instructor(ID, first_name, middle_initial, last_name, street_number, street_name, apt_number, city, state, zip, phone_number, date_of_birth, age)`. 
    - \( \text{phone_number} \) is a multivalued attribute, denoted by \{ phone_number \}.
    - \( \text{date_of_birth} \) is a scalar value.
    - \( \text{age} \) is a calculated attribute, denoted by \( \text{age} () \).
A multivalued attribute $M$ of an entity $E$ is represented by a separate schema $EM$.

- Schema $EM$ has attributes corresponding to the primary key of $E$ and an attribute corresponding to multivalued attribute $M$.

- Example: Multivalued attribute `phone_number` of `instructor` is represented by a schema:
  
  $\text{inst\_phone} = (\text{ID}, \text{phone\_number})$

- Each value of the multivalued attribute maps to a separate tuple of the relation on schema $EM$.
  
  - For example, an `instructor` entity with primary key 22222 and phone numbers 456-7890 and 123-4567 maps to two tuples: (22222, 456-7890) and (22222, 123-4567)
Special case: entity *time_slot* has only one attribute other than the primary-key attribute, and that attribute is multivalued

- Optimization: Don’t create the relation corresponding to the entity, just create the one corresponding to the multivalued attribute
- \( \text{time_slot}(\text{time_slot_id, day, start_time, end_time}) \)
- Caveat: *time_slot* attribute of *section* (from \( \text{sec_time_slot} \)) cannot be a foreign key due to this optimization
Design Issues

■ Use of entity sets vs. attributes

- Designing phone as an entity allow for primary key constraints for phone
- Designing phone as an entity allow phone numbers to be used in relationships with other entities (e.g., student)
- Use of phone as an entity allows extra information about phone numbers
Design Issues

- **Use of entity sets vs. relationship sets**
  - Possible guideline is to designate a relationship set to describe an action that occurs between entities.
  - Possible hint: the relationship only relates entities, but does not have an existence by itself. E.g., `hasAddress`: `(department-address)`

```plaintext
section
sec_id
semester
year

registration
...
...
...

student
ID
name
tot_cred

section_reg
student_reg
```
Design Issues

- **Binary versus n-ary relationship sets**
  - Although it is possible to replace any nonbinary (n-ary, for \( n > 2 \)) relationship set by a number of distinct binary relationship sets + an artificial entity set, a n-ary relationship set shows more clearly that several entities participate in a single relationship.

- **Placement of relationship attributes**
  - e.g., attribute `date` as attribute of `advisor` or as attribute of `student`
  - Does not work for \( N-M \) relationships!
Some relationships that appear to be non-binary may be better represented using binary relationships

- E.g., A ternary relationship parents, relating a child to his/her father and mother, is best replaced by two binary relationships, father and mother
  - Using two binary relationships allows partial information (e.g., only mother being known)

- But there are some relationships that are naturally non-binary
  - Example: proj_guide
Converting Non-Binary Relationships to Binary Form

In general, any non-binary relationship can be represented using binary relationships by creating an artificial entity set.

- Replace $R$ between entity sets A, B and C by an entity set $E$, and three relationship sets:
  1. $R_A$, relating $E$ and $A$
  2. $R_B$, relating $E$ and $B$
  3. $R_C$, relating $E$ and $C$

- Create a special identifying attribute for $E$

- Add any attributes of $R$ to $E$

- For each relationship $(a_i, b_i, c_i)$ in $R$, create
  1. a new entity $e_i$ in the entity set $E$
  2. add $(e_i, a_i)$ to $R_A$
  3. add $(e_i, b_i)$ to $R_B$
  4. add $(e_i, c_i)$ to $R_C$

![Diagram](image-url)
Also need to translate constraints

- Translating all constraints may not be possible
- There may be instances in the translated schema that cannot correspond to any instance of \( R \)
  
  Exercise: *add constraints to the relationships \( R_A, R_B \) and \( R_C \) to ensure that a newly created entity corresponds to exactly one entity in each of entity sets \( A, B \) and \( C \)*

- We can avoid creating an identifying attribute by making \( E \) a weak entity set (described shortly) identified by the three relationship sets
Converting Non-Binary Relationships: Is the New Entity Set E Necessary?

- Yes, because a non-binary relationship stores more information than any number of binary relationships.
  - Consider again the example (a) below.
  - Replace R with three binary relationships:
    1. $R_{AB}$, relating A and B
    2. $R_{BC}$, relating B and C
    3. $R_{AC}$, relating A and C
  - For each relationship $(a_i, b_i, c_i)$ in R, create
    1. add $(a_i, b_i)$ to $R_{AB}$
    2. add $(b_i, c_i)$ to $R_{BC}$
    3. add $(a_i, c_i)$ to $R_{AC}$
  - Consider $R = \text{order}$, $A = \text{supplier}$, $B = \text{item}$, $C = \text{customer}$
    
    (Gunnar, chainsaw, Bob) – Bob ordered a chainsaw from Gunnar
    ->
    
    (Gunnar, chainsaw), (chainsaw, Bob), (Gunnar, Bob)

    Gunnar supplies chainsaws, Bob ordered a chainsaw, Bob ordered something from Gunnar. E.g., we do not know what Bob ordered from Gunnar.
ER-model to Relational Summary

**Rule 1) Strong entity E**
- Create relation with attributes of E
- Primary key is equal to the PK of E

**Rule 2) Weak entity W identified by E through relationship R**
- Create relation with attributes of W and R and PK(E).
- Set PK to discriminator attributes combined with PK(E). PK(E) is a foreign key to E.

**Rule 3) Binary relationship R between A and B: one-to-one**
- If no side is total add PK of A to as foreign key in B or the other way around. Add any attributes of the relationship R to A respective B.
- If one side is total add PK of the other-side as foreign key. Add any attributes of the relationship R to the total side.
- If both sides are total merge the two relation into a new relation E and choose either PK(A) as PK(B) as the new PK. Add any attributes of the relationship R to the new relation E.
ER-model to Relational Summary (Cont.)

■ **Rule 4)** Binary relationship R between A and B: one-to-many/many-to-one
  - Add PK of the “one” side as foreign key to the “many” side.
  - Add any attributes of the relationship R to the “many” side.

■ **Rule 5)** Binary relationship R between A and B: many-to-many
  - Create a new relation R.
  - Add PK’s of A and B as attributes + plus all attributes of R.
  - The primary key of the relationship is PK(A) + PK(B). The PK attributes of A/B form a foreign key to A/B.

■ **Rule 6)** N-ary relationship R between $E_1 \ldots E_n$
  - Create a new relation.
  - Add all the PK’s of $E_1 \ldots E_n$. Add all attributes of R to the new relation.
  - The primary key or R is PK($E_1$) … PK($E_n$). Each PK($E_i$) is a foreign key to the corresponding relation.
Rule 7) Entity $E$ with multi-valued attribute $A$

- Create new relation. Add $A$ and $PK(E)$ as attributes.
- $PK$ is all attributes. $PK(E)$ is a foreign key.
Rule 1) Strong Entities
- department\(\text{dept\_name}, \text{building}, \text{budget}\)
- instructor\(\text{ID}, \text{name}, \text{salary}\)
- student\(\text{ID}, \text{name}, \text{tot\_cred}\)
- course\(\text{course\_id}, \text{title}, \text{credits}\)
- time\_slot\(\text{time\_slot\_id}\)
- classroom\(\text{building}, \text{room\_number}, \text{capacity}\)

Rule 2) Weak Entities
- section\(\text{course\_id}, \text{sec\_id}, \text{semester}, \text{year}\)
Rule 3) Relationships one-to-one
- None exist

Rule 4) Relationships one-to-many
- department(dept_name, building, budget)
- instructor(ID, name, salary, dept_name)
- student(ID, name, tot_cred, dept_name, instr_ID)
- course(course_id, title, credits, dept_name)
- time_slot(time_slot_id)
- classroom(building, room_number, capacity)
- section(course_id, sec_id, semester, year, room_building, room_number, time_slot_id)
Translate the University ER-Model

- **Rule 5) Relationships many-to-many**
  - department\( (\text{dept\_name}, \text{building}, \text{budget}) \)
  - instructor\( (\text{ID}, \text{name}, \text{salary}, \text{dept\_name}) \)
  - student\( (\text{ID}, \text{name}, \text{tot\_cred}, \text{dept\_name}, \text{instr\_ID}) \)
  - course\( (\text{course\_id}, \text{title}, \text{credits}, \text{dept\_name}) \)
  - time_slot\( (\text{time\_slot\_id}) \)
  - classroom\( (\text{building}, \text{room\_number}, \text{capacity}) \)
  - section\( (\text{course\_id}, \text{sec\_id}, \text{semester}, \text{year}, \text{room\_building}, \text{room\_number}, \text{time\_slot\_id}) \)
  - prereq\( (\text{course\_id}, \text{prereq\_id}) \)
  - teaches\( (\text{ID}, \text{course\_id}, \text{sec\_id}, \text{semester}, \text{year}) \)
  - takes\( (\text{ID}, \text{course\_id}, \text{sec\_id}, \text{semester}, \text{year}, \text{grade}) \)

- **Rule 6) N-ary Relationships**
  - none exist
Rule 7) Multivalued attributes

- department (dept_name, building, budget)
- instructor (ID, name, salary, dept_name)
- student (ID, name, tot_cred, dept_name, instr_ID)
- course (course_id, title, credits, dept_name)
- time_slot (time_slot_id)
- time_slot_day (time_slot_id, start_time, end_time)
- classroom (building, room_number, capacity)
- section (course_id, sec_id, semester, year, room_building, room_number, time_slot_id)
- prereq (course_id, prereq_id)
- teaches (ID, course_id, sec_id, semester, year)
- takes (ID, course_id, sec_id, semester, year, grade)
Extended ER Features
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., instructor “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

person
  ID
  name
  address

employee
  salary

student
  tot_credits

instructor
  rank

secretary
  hours_per_week
**Extended ER Features: 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 and Generalization (Cont.)

- Can have multiple specializations of an entity set based on different features.

- E.g., `permanent_employee` vs. `temporary_employee`, in addition to `instructor` vs. `secretary`

- Each particular employee would be:
  - a member of one of `permanent_employee` or `temporary_employee`,
  - and also a member of one of `instructor`, `secretary`

- The ISA relationship also referred to as `superclass - subclass` relationship
Design Constraints on a 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; 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.
  - Disjoint
    - an entity can belong to only one lower-level entity set
    - Noted in E-R diagram by having multiple lower-level entity sets link to the same triangle
  - Overlapping
    - an entity can belong to more than one lower-level entity set
Specialization Example

Disjoint, employees are either instructors or secretaries

Overlapping, a person can be both an employee and a student
Design Constraints on a Specialization/Generalization (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
Consider the ternary relationship `proj_guide`, which we saw earlier.

Suppose we want to record evaluations of a student by a guide on a project.
Aggregation (Cont.)

- Relationship sets `eval_for` and `proj_guide` represent overlapping information
  - Every `eval_for` relationship corresponds to a `proj_guide` relationship
  - However, some `proj_guide` relationships may not correspond to any `eval_for` relationships
    - So we can’t discard the `proj_guide` 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:

- A student is guided by a particular instructor on a particular project
- A student, instructor, project combination may have an associated evaluation
Representing Specialization via Schemas

Method 1:

- Form a relation schema for the higher-level entity
- Form a relation schema for each lower-level entity set, include primary key of higher-level entity set and local attributes

<table>
<thead>
<tr>
<th>schema</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>ID, name, street, city</td>
</tr>
<tr>
<td>student</td>
<td>ID, tot_cred</td>
</tr>
<tr>
<td>employee</td>
<td>ID, salary</td>
</tr>
</tbody>
</table>

- Drawback: getting information about, an employee requires accessing two relations, the one corresponding to the low-level schema and the one corresponding to the high-level schema
Method 2:

- Form a single relation schema for each entity set with all local and inherited attributes.

<table>
<thead>
<tr>
<th>schema</th>
<th>attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>person</td>
<td>ID, name, street, city</td>
</tr>
<tr>
<td>student</td>
<td>ID, name, street, city, tot_cred</td>
</tr>
<tr>
<td>employee</td>
<td>ID, name, street, city, salary</td>
</tr>
</tbody>
</table>

- If specialization is total, the schema for the generalized entity set (person) not required to store information.
  - Can be defined as a "view" relation containing union of specialization relations.
  - But explicit schema may still be needed for foreign key constraints.

- Drawback: name, street and city may be stored redundantly for people who are both students and employees.
Representing Specialization as Schemas (Cont.)

- **Method 3:**
  - Form a single relation schema for each entity set with all local and inherited attributes
    - For total and disjoint specialization add a single “type” attribute that stores the type of an entity
      - Person ID, type, name, street, city, total_credits, salary
    - For partial and/or overlapping specialization add multiple boolean “type” attributes
      - Person ID, isEmployee, isStudent, name, street, city, total_credits, salary
  - Drawback: large number of NULL values, potentially large relation
Schemas Corresponding to Aggregation

To represent aggregation, create a schema containing

- primary key of the aggregated relationship,
- the primary key of the associated entity set
- any descriptive attributes
For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema

\[ \text{eval_for}(s\_ID, project\_id, i\_ID, evaluation\_id) \]
Rule 8) Specialization of E into S₁, … , Sₙ (method 1)
- Create a relation for E with all attributes of E. The PK of E is the PK.
- For each Sᵢ create a relation with PK(E) as PK and foreign key to relation for E. Add all attributes of Sᵢ that do not exist in E.

Rule 9) Specialization of E into S₁, … , Sₙ (method 2)
- Create a relation for E with all attributes of E. The PK of E is the PK.
- For each Sᵢ create a relation with PK(E) as PK and foreign key to relation for E. Add all attributes of Sᵢ.

Rule 10) Specialization of E into S₁, … , Sₙ (method 3)
- Create a new relation with all attributes from E and S₁, … , Sₙ.
- Add single attribute type or a boolean type attribute for each Sᵢ.
- The primary key is PK(E)
Rule 11) Aggregation: Relationship $R_1$ relates entity sets $E_1$, ..., $E_n$. This is related by relationship A to an entity set B

- Create a relation for A with attributes $PK(E_1)$, ..., $PK(E_n)$ + all attributes from A + $PK(B)$. $PK$ are all attributes except the ones from A
ER Design Decisions

- 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.
How about doing another ER design interactively on the board?
Summary of Symbols Used in E-R Notation

- **E**: entity set
- **R**: relationship set
- **E** → **R**: identifying relationship set for weak entity set
- **R** → **E**: total participation of entity set in relationship

**Attributes**:
- Simple (A1)
- Composite (A2)
- Multivalued (A3)
- Derived (A4)
- Primary key
- Discriminating attribute of weak entity set

**Cardinality**:
- One-to-one
- Many-to-one
- Many-to-many

** ISA**: generalization or specialization
Symbols Used in ER Notation (Cont.)

- **many-to-many relationship**: 
  - \( R \)

- **one-to-one relationship**: 
  - \( R \)

- **role name**: 
  - \( R \rightarrow E \)

- **role indicator**: 
  - \( R \rightarrow E \)

- **relationship set**: 
  - \( R \rightarrow E \)

- **entity set**: 
  - \( E \)

- **identifying relationship set**: 
  - \( R \rightarrow E \)

- **primary key**: 
  - \( R \rightarrow E \)

- **many-to-one relationship**: 
  - \( R \rightarrow E \)

- **cardinality limits**: 
  - \( R \rightarrow E \)

- **ISA: generalization or specialization**: 
  - \( E \rightarrow E \)

- **disjoint generalization**: 
  - \( E \rightarrow E \)

- **total (disjoint) generalization**: 
  - \( E \rightarrow E \)

- **total participation of entity set in relationship**: 
  - \( \text{total} \)

- **simple (A1)**, **composite (A2)** and **multivalued (A3)**, **derived (A4)** attributes:
  - \( \text{total} \)

- **ISA**: 
  - \( E \rightarrow E \)

- ** ISA: generalization or specialization**: 
  - \( E \rightarrow E \)

- **disjoint generalization**: 
  - \( E \rightarrow E \)
Alternative ER Notations

- Chen, IDE1FX, ...

entity set E with simple attribute A1, composite attribute A2, multivalued attribute A3, derived attribute A4, and primary key A1

weak entity set  

generalization  

ISA  

total generalization  

ISA
Alternative ER Notations

Chen

IDE1FX (Crows feet notation)

many-to-many relationship

one-to-one relationship

many-to-one relationship

participation in R: total (E1) and partial (E2)
UML: Unified Modeling Language

UML has many components to graphically model different aspects of an entire software system

UML Class Diagrams correspond to E-R Diagram, but several differences.
**ER vs. UML Class Diagrams**

### ER Diagram Notation

**Entity with Attributes**

<table>
<thead>
<tr>
<th>E</th>
<th>A1</th>
<th>M1(0)</th>
</tr>
</thead>
</table>

- E: Entity
- A1: Simple attribute
- M1(0): Multivalued, derived attribute

### Equivalent in UML

**Class with Attributes**

| E   | –A1  | +M1(0) |

- E: Class
- –A1: Private attribute
- +M1(0): Public attribute

### Binary Relationship

- E1: Role1
- R: Relationship
- E2: Role2

**Relationship Attributes**

- E1: Role1
- R: Relationship
- E2: Role2

### Cardinality Constraints

- E1: Role1
- R: Relationship
- E2: Role2

**Note reversal of position in cardinality constraint depiction**
ER vs. UML Class Diagrams

ER Diagram Notation

Equivalent in UML

*Generalization can use merged or separate arrows independent of disjoint/overlapping
UML Class Diagrams (Cont.)

- Binary relationship sets are represented in UML by just drawing a line connecting the entity sets. The relationship set name is written adjacent to the line.

- The role played by an entity set in a relationship set may also be specified by writing the role name on the line, adjacent to the entity set.

- The relationship set name may alternatively be written in a box, along with attributes of the relationship set, and the box is connected, using a dotted line, to the line depicting the relationship set.
Recap

- ER-model
  - Entities
    - Strong
    - Weak
  - Attributes
    - Simple vs. Composite
    - Single-valued vs. Multi-valued
  - Relationships
    - Degree (binary vs. N-ary)
  - Cardinality constraints
  - Specialization/Generalization
    - Total vs. partial
    - Disjoint vs. overlapping
  - Aggregation
Recap Cont.

- ER-Diagrams
  - Alternative notations
- UML-Diagrams
- Design decisions
  - Multi-valued attribute vs. entity
  - Entity vs. relationship
  - Binary vs. N-ary relationships
  - Placement of relationship attributes
  - Total 1-1 vs. single entity
- ER to relational model
  - Translation rules
End of Chapter 7

Partially taken from
Klaus R. Dittrich

modified from:
Database System Concepts, 6th Ed.
©Silberschatz, Korth and Sudarshan
See www.db-book.com for conditions on re-use
Outline

- Introduction
- Relational Data Model
- Formal Relational Languages (relational algebra)
- SQL - Advanced
- **Database Design – Database modelling**
- Transaction Processing, Recovery, and Concurrency Control
- Storage and File Structures
- Indexing and Hashing
- Query Processing and Optimization
Figure 7.01

instructor

<table>
<thead>
<tr>
<th>76766</th>
<th>Crick</th>
</tr>
</thead>
<tbody>
<tr>
<td>45565</td>
<td>Katz</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
</tr>
</tbody>
</table>

student

<table>
<thead>
<tr>
<th>98988</th>
<th>Tanaka</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Shankar</td>
</tr>
<tr>
<td>00128</td>
<td>Zhang</td>
</tr>
<tr>
<td>76543</td>
<td>Brown</td>
</tr>
<tr>
<td>76653</td>
<td>Aoi</td>
</tr>
<tr>
<td>23121</td>
<td>Chavez</td>
</tr>
<tr>
<td>44553</td>
<td>Peltier</td>
</tr>
</tbody>
</table>
Figure 7.02

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766 Crick</td>
<td>98988 Tanaka</td>
</tr>
<tr>
<td>45565 Katz</td>
<td>12345 Shankar</td>
</tr>
<tr>
<td>10101 Srinivasan</td>
<td>00128 Zhang</td>
</tr>
<tr>
<td>98345 Kim</td>
<td>76543 Brown</td>
</tr>
<tr>
<td>76543 Singh</td>
<td>76653 Aoi</td>
</tr>
<tr>
<td>22222 Einstein</td>
<td>23121 Chavez</td>
</tr>
<tr>
<td></td>
<td>44553 Peltier</td>
</tr>
</tbody>
</table>
Figure 7.03

instructor

76766 Crick
45565 Katz
10101 Srinivasan
98345 Kim
76543 Singh
22222 Einstein

student

98988 Tanaka
12345 Shankar
00128 Zhang
76543 Brown
76653 Aoi
23121 Chavez
44553 Peltier

3 May 2008
10 June 2007
12 June 2006
6 June 2009
30 June 2007
31 May 2007
4 May 2006
Figure 7.04

composite attributes

name

first_name  middle_initial  last_name

component attributes

street

city

state

postal_code

street_number

street_name

apartment_number
Figure 7.05

(a) A \[ \begin{align*} a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ a_4 \end{align*} \]

(b) A \[ \begin{align*} a_1 \\ b_1 \\ a_2 \\ b_2 \\ a_3 \\ b_3 \\ a_4 \\ b_4 \\ a_5 \end{align*} \]
Figure 7.08

- Instructor:
  - ID
  - Name
  - Salary

- Advisor:
  - Date

- Student:
  - ID
  - Name
  - Tot_cred
Figure 7.10

```
<table>
<thead>
<tr>
<th>instructor</th>
<th>student</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>ID</td>
</tr>
<tr>
<td>name</td>
<td>name</td>
</tr>
<tr>
<td>salary</td>
<td>tot_cred</td>
</tr>
</tbody>
</table>

0..*  advisor  1..1
```
Figure 7.11

```
<table>
<thead>
<tr>
<th>instructor</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
</tr>
<tr>
<td>name</td>
</tr>
<tr>
<td>first_name</td>
</tr>
<tr>
<td>middle_initial</td>
</tr>
<tr>
<td>last_name</td>
</tr>
<tr>
<td>address</td>
</tr>
<tr>
<td>street</td>
</tr>
<tr>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
</tr>
<tr>
<td>apt_number</td>
</tr>
<tr>
<td>city</td>
</tr>
<tr>
<td>state</td>
</tr>
<tr>
<td>zip</td>
</tr>
<tr>
<td>{ phone_number }</td>
</tr>
<tr>
<td>date_of_birth</td>
</tr>
<tr>
<td>age ( )</td>
</tr>
</tbody>
</table>
```
Figure 7.12

course
course_id
title
credits

prereq
prereq_id
course_id
Figure 7.13

- course
  - course_id
  - title
  - credits
- course_id: course_id
- prereq_id: prereq_id
- prereq

The diagram illustrates a relationship between courses and prerequisites, with the course_id linking the course table to the prereq table.
Figure 7.15

The figure illustrates a database schema with the following entities and relationships:

- **course_dept**
  - course_dept
  - course_id
  - title
  - credits
  - course_dept
  - prereq
  - prereq_id

- **inst_dept**
  - ID
  - name
  - salary

- **dept_name**
  - building
  - budget

- **stud_dept**
  - ID
  - name
  - tot_cred

- **teaches**
  - course_id
  - title
  - credits

- **sec_course**
  - sec_id
  - semester
  - year

- **section**
  - sec_id
  - semester
  - year

- **sec_time_slot**
  - time_slot_id
  - day
  - start_time
  - end_time

- **classroom**
  - building
  - room_number
  - capacity

- **advisor**
  - student

- **instructor**
  - ID
  - name
  - salary

- **student**
  - ID
  - name
  - tot_cred

- **grade**

The diagram shows the relationships between these entities, indicating how courses, instructors, and students interact within the database schema.
Figure 7.17

(a) 

(b) 

instructor

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>salary</th>
<th>phone_number</th>
</tr>
</thead>
</table>

instructor

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>salary</th>
</tr>
</thead>
</table>

inst_phone

phone

<table>
<thead>
<tr>
<th>phone_number</th>
<th>location</th>
</tr>
</thead>
</table>

(a) (b)
Figure 7.18

The diagram shows a database schema with entities and relationships:

- **section_reg** entity with attributes: sec_id, semester, year.
- **registration** entity with attributes: ...
- **student_reg** entity with attributes: ...
- **student** entity with attributes: ID, name, tot_cred.
- **section** entity with attributes: sec_id, semester, year.

The entities are connected with relationships indicated by lines between them.
Figure 7.19

(a) (b)
Figure 7.20

<table>
<thead>
<tr>
<th>Instructor</th>
<th>Student</th>
</tr>
</thead>
<tbody>
<tr>
<td>76766</td>
<td>Crick</td>
</tr>
<tr>
<td>45565</td>
<td>Katz</td>
</tr>
<tr>
<td>10101</td>
<td>Srinivasan</td>
</tr>
<tr>
<td>98345</td>
<td>Kim</td>
</tr>
<tr>
<td>76543</td>
<td>Singh</td>
</tr>
<tr>
<td>22222</td>
<td>Einstein</td>
</tr>
<tr>
<td>98988</td>
<td>Tanaka</td>
</tr>
<tr>
<td>12345</td>
<td>Shankar</td>
</tr>
<tr>
<td>00128</td>
<td>Zhang</td>
</tr>
<tr>
<td>76543</td>
<td>Brown</td>
</tr>
<tr>
<td>76653</td>
<td>Aoi</td>
</tr>
<tr>
<td>23121</td>
<td>Chavez</td>
</tr>
<tr>
<td>44553</td>
<td>Peltier</td>
</tr>
</tbody>
</table>

instructor

student

©Silberschatz, Korth and Sudarshan
Figure 7.21

- **person**
  - ID
  - name
  - address

- **employee**
  - salary

- **student**
  - tot_credits

- **instructor**
  - rank

- **secretary**
  - hours_per_week
Figure 7.22

The diagram illustrates the relationships between project, instructor, student, proj_guide, eval_for, and evaluation. The project is connected to both instructor and student through proj_guide and eval_for, respectively.
Figure 7.23

- `project`
- `instructor`
- `student`
- `proj_guide`
- `eval_for`
- `evaluation`
Figure 7.24

- **Entity set (E)**
- **Relationship set (R)**
- **Identifying relationship set for weak entity set**
- **Total participation of entity set in relationship**
- **Many-to-many relationship**
- **One-to-one relationship**
- **Role indicator**
- **Total (disjoint) generalization**
- **ISA: Generalization or specialization**

**Attributes:**
- Simple (A1)
- Composite (A2)
- Multivalued (A3)
- Derived (A4)

** ISA:**
- Generalization or specialization

**Role indicator:**
- Total
- Disjoint generalization

**Cardinality limits:**
- Many-to-one relationship

**Primary key:**
- A1
entity set E with simple attribute A1, composite attribute A2, multivalued attribute A3, derived attribute A4, and primary key A1

many-to-many relationship

one-to-one relationship

many-to-one relationship

participation in R: total (E1) and partial (E2)

weak entity set

generalization

total generalization

ISA
Figure 7.26

ER Diagram Notation

<table>
<thead>
<tr>
<th>E</th>
<th>entity with attributes (simple, composite, multivalued, derived)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A1</td>
<td></td>
</tr>
<tr>
<td>M10</td>
<td></td>
</tr>
</tbody>
</table>

Equivalent in UML

| E    | class with simple attributes and methods (attribute prefixes: + = public, - = private, # = protected) |

A1

E1

role1

R

role2

E2

relationship

attributes

E1

0.. * R 0.. 1

cardinality constraints

E1

0.. 1 R 0.. *

E2

role1

R

role2

E2

n-ary relationships

E1

R

E2

E3

overlapping generalization

E1

E2

E3

disjoint generalization

E1

E2

E3

overlapping

E1

E2

E3

disjoint
Figure 7.27

(a) (b) (c)
Figure 7.28
Figure 7.29