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

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.

Requirement Analysis Example Music Collection

- Let's do it!
This page discusses database design concepts. It begins with a discussion on formalizing a model by developing a conceptual model. The page outlines the steps involved in the design process, including requirement analysis, conceptual modeling, and logical modeling (possibly automated).

The page also delves into conceptual modeling, explaining that 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. Examples include specific persons, companies, events, plants. Entities have attributes, such as names and addresses. An entity set is a set of entities of the same type that share the same properties. Examples include all persons, companies, trees, holidays.

Relationship sets are also discussed, mentioning that a relationship is an association among several entities. An example is given: 44553 (Peltier) advises 22222 (Einstein).

Entity sets are introduced, with examples such as:
- Instructor ID: 76766, Instructor Name: Crick
- Student ID: 98988, Student Name: Tanaka
- Instructor ID: 10101, Student Name: Shankar
- Student ID: 12345, Instructor Name: Srinivasan
- Instructor ID: 98345, Student Name: Kim
- Student ID: 76543, Instructor Name: Zhang
- Instructor ID: 76543, Student Name: Brown
- Student ID: 76653, Student Name: Aoi
- Student ID: 23121, Instructor Name: Chavez
- Instructor ID: 44553, Student Name: Peltier

The page further explains relationship sets, providing an example of an advisor relationship set: 76766 Crick advises 22222 Einstein.

Overall, the page provides a comprehensive overview of database design concepts, focusing on the importance of modeling and formalizing databases to ensure they meet the needs of the application or system they support.
Relationship Sets (Cont.)

- An attribute can also be a 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

- A binary relationship involves 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:
    - instructor = (ID, name, street, city, salary)
    - course = (course_id, title, credits)
- Domain – the set of permitted values for each attribute
- Attribute types:
  - Simple and composite attributes.
  - Single-valued and multivalued attributes
    - Example: multivalued attribute: phone_numbers
  - Derived attributes
    - Can be computed from other attributes
    - Example: age, given date_of_birth

Composite Attributes

- composite attributes
  - street_number
  - street_name
  - apartment_number

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

- 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 Example

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

Many to one

Many to many

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

Mapping Cardinalities Example

Employee

Department

Student

Course

(a)

(b)

Many to one

Many to many

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

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
0:1
1:0:1
0:1-0:1
1-N
0:1-N
0:1-0:N
1:N
0:N-1
0:N-0:1
N-1
N-0:1
0:1-N
N-N
0:N-0:M

Typical Notation
(0:1) → (1:N)

Keys

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 advisor
  - 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 primary key in case of more than one candidate key

Keys for Relationship Sets Cont.

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

<table>
<thead>
<tr>
<th>composite</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>

<table>
<thead>
<tr>
<th>derived</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>
Relationship Sets with Attributes

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

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.

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
  - a student is associated with at most one instructor via 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

Many-to-One Relationships
- 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
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

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

Alternative Notation for Cardinality Limits

- Cardinality limits can also express participation constraints

Cardinality Constraints on Ternary Relationship

- 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 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
  - 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

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

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.)

- 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.

E-R Diagram for a University Enterprise
Reduction to Relational 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.

Representing Entity Sets With Simple Attributes

- A strong entity set reduces to a schema with the same attributes student(ID, name, tot_cread)
- 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)

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_code, date_of_birth)

Multivalued Attributes (Cont.)

- 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.
  - time_slot(day, start_time, end_time)
  - Caveat: time_slot attribute of section (from sec_time_slot) cannot be a foreign key due to this optimization.

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)

- 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 many 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!
Binary Vs. Non-Binary 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. R₁, relating E and A
    2. R₂, relating E and B
    3. R₃, relating E and C
  - Create a special identifying attribute for E.
  - Add any attributes of R to E.
  - For each relationship (a, b, c) in R, create:
    1. a new entity e in the entity set E
    2. add (e, a) to R₁
    3. add (e, b) to R₂
    4. add (e, c) to R₃

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. R₁, relating A and B
    2. R₂, relating B and C
    3. R₃, relating A and C
  - For each relationship (a, b, c) in R, create:
    1. add (a, b) to R₁
    2. add (b, c) to R₂
    3. add (a, c) to R₃
  - Consider R = order, A = supplier, B = item, C = 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 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 PKs of A and B as attributes and 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 E₁... Eₙ
  - Create a new relation.
  - Add all the PKs of E₁... Eₙ. Add all attributes of R to the new relation.
  - The primary key or R is PK(E₁)... PK(Eₙ). Each PK(E) is a foreign key to the corresponding relation.
Translate the University ER-Model

- Rule 1) Strong Entities
  - department (dept_name, building, budget)
  - instructor (ID, name, salary)
  - student (ID, name, tot_cred)
  - course (course_id, title, credits)
  - classroom (building, room_number, capacity)
- Rule 2) Weak Entities
  - sec_course (course_id, sec_id, semester, 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)
- Rule 5) Relationships many-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)
  - prereq (course_id, prereq_id)
- Rule 6) N-ary Relationships
  - None exist

Translate the University ER-Model

- Rule 1) Strong Entities
  - 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)
  - classroom (building, room_number, capacity)
  - section (course_id, sec_id, semester, year)
  - time_slot (time_slot_id)
  - prereq (course_id, prereq_id)
  - teaches (ID, course_id, sec_id, semester, year)
  - takes (ID, course_id, sec_id, semester, year)
- Rule 2) Weak Entities
  - courseDept
  - secClass
  - classroom
  - section
  - time_slot
- 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)
- Rule 5) Relationships many-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)
  - prereq (course_id, prereq_id)
- Rule 6) N-ary Relationships
  - None exist
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

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
  - Overlapping
    - an entity can belong to more than one lower-level entity set

Specialization Example

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
  - 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

Employee

Student

Instructor

Secretary

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

Aggregation

- 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 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

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
    - 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
Representing Specialization as Schemas (Cont.)

- Method 2:
  - Form a single relation schema for each entity set with all local and inherited attributes.
  - Schemas corresponding to aggregation
  - 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.

- Method 3:
  - Form a single relation schema for each entity set with all local and inherited attributes.
  - For partial and overlapping specialization add multiple boolean "type" attributes.
  - 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

ER-model to Relational Summary (Cont.)

- Rule 8) Specialization of $E$ into $S_1, ..., S_n$ (method 1)
  - Create a relation for $E$ with all attributes of $E$. The PK of $E$ is the PK.
  - For each $S_i$ create a relation with PK($E$) as PK and foreign key to relation for $E$. Add all attributes of $S_i$ that do not exist in $E$.

- Rule 9) Specialization of $E$ into $S_1, ..., S_n$ (method 2)
  - Create a relation for $E$ with all attributes of $E$. The PK of $E$ is the PK.
  - For each $S_i$ create a relation with PK($E$) as PK and foreign key to relation for $E$. Add all attributes of $S_i$.

- Rule 10) Specialization of $E$ into $S_1, ..., S_n$ (method 3)
  - Create a new relation with all attributes from $E$ and $S_1, ..., S_n$.
  - Add single attribute type or a boolean attribute for each $S_i$.
  - The primary key is PK($E$).

Representing Specialization as Schemas (Cont.)

- For total and disjoint specialization add a single "type" attribute that stores the type of an entity.

Schemas Corresponding to Aggregation (Cont.)

For example, to represent aggregation manages between relationship works_on and entity set manager, create a schema.

```
<table>
<thead>
<tr>
<th>manager</th>
<th>project_id</th>
<th>eval_for</th>
</tr>
</thead>
</table>
```

ER-model to Relational Summary (Cont.)

- Rule 11) Aggregation: Relationship $R$ 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($E$) + PK($E$) + 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.

Summary of Symbols Used in E-R Notation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Entity set</td>
</tr>
<tr>
<td>R</td>
<td>Relationship set</td>
</tr>
<tr>
<td>K</td>
<td>Primary key</td>
</tr>
<tr>
<td>A</td>
<td>Attribute</td>
</tr>
<tr>
<td>A1, A2, A3, A4</td>
<td>Attributes: simple (A1), composite (A2) and multivalued (A3) derived (A4)</td>
</tr>
</tbody>
</table>

Alternative ER Notations

- Chen, IDEAFX, ...

Symbols Used in ER Notation (Cont.)

- Many-to-many relationship
- One-to-one relationship
- Cardinality limits
- ISA: generalization or specialization
- Disjoint generalization

Alternative ER Notations

- Chen
- IDEAFX (Crows feet notation)
**UML**

- **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**

<table>
<thead>
<tr>
<th>ER Diagram Notation</th>
<th>Equivalent in UML</th>
</tr>
</thead>
<tbody>
<tr>
<td>E1</td>
<td>E</td>
</tr>
<tr>
<td>A1</td>
<td>A1</td>
</tr>
<tr>
<td>M1</td>
<td>M1</td>
</tr>
<tr>
<td>E2</td>
<td>role1</td>
</tr>
<tr>
<td>E3</td>
<td>role2</td>
</tr>
</tbody>
</table>

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.

*Generalization can use merged or separate arrows independent of disjoint/overlapping

**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

- **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

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

Partially taken from
Klaus R. Dittrich

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

<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>44553 Peltier</td>
<td></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>44553 Peltier</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.03

<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>44553 Peltier</td>
<td></td>
</tr>
</tbody>
</table>

Figure 7.04

<table>
<thead>
<tr>
<th>Composite Attributes</th>
<th>Component Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>street_number</td>
</tr>
<tr>
<td>middle_initial</td>
<td>street_number</td>
</tr>
<tr>
<td>last_name</td>
<td>street_name</td>
</tr>
<tr>
<td>street</td>
<td>state</td>
</tr>
<tr>
<td>city</td>
<td>postal_code</td>
</tr>
<tr>
<td>street_number</td>
<td>street_number</td>
</tr>
<tr>
<td>street_name</td>
<td>apartment_number</td>
</tr>
</tbody>
</table>

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

© Silberschatz, Korth and Sudarshan
Figure 7.05

Figure 7.06

Figure 7.07

Figure 7.08

Figure 7.09

Figure 7.10
ER Diagram Notation Equivalent in UML

entity set E1

relationship set R

e2

e1

role1

0.. *

e2

e3

e1

E2

role indicator

total (disjoint) participation in E1 total (disjoint) participation in E1

relationship set

many-to-many relationship

one-to-one relationship

many-to-one relationship

weak entity set

discriminating attribute

primary key

ISA: generalization

n-ary constraints

cardinality

relationship

binary relationship

class with simple attributes and methods...

URL

address

name

price

year

ISBN

written_by

author

book

shopping_basket

basket_id

warehouse

basket

title

year

price

code

address

phone

URL

customer

e-mail

name

address

phone

author

book

published_by

publisher

e-mail

name

address

phone

URL

customer

shopping_basket

basket_of

title

year

price

basket_id

warehouse

basket

title

year

price

basket_id

warehouse

basket