Entity-Relationship and Enhanced Entity-Relationship Conceptual Data Models (Chapters 7 & Sections 2.1 to 2.3)

E-R MODEL

- E-R model is based on a real world that consists of entities and relationships. It consists of two primitives:
  1. Entities: objects that exist and are distinguishable from one another. These objects might be abstract, e.g., SF-49ers, Joe, Bob, etc.
  2. Relationships: two or more entities may participate in a relationship, e.g., Joe is a fan of SF-49ers, etc.
- An entity is represented by a set of attributes, e.g., Joe might have a social-security number, a phone number, a spouse, and a salary
- An attribute is a function that maps an entity into a domain, e.g., Joe's social-security number maps Joe into the domain of integers. An attribute is represented as an
- A set of entities of the same type is termed an entity set, e.g., football teams. Entity sets do not need to be disjoint, e.g., students, employees, people. An entity set is represented as a

RELATIONSHIPS

- Relationships of the same type are contained in a relationship set. More formally, if \( E_1, E_2, \ldots, E_n \) are entity sets, then a relationship set \( R \) is a subset of \( \{(e_1, e_2, \ldots, e_n) | e_i \in E_i \text{ for } i = 1, 2, \ldots, n \} \) where \( (e_1, e_2, \ldots, e_n) \) is a relationship. A relationship set is represented as a
- A relationship may involve two entities, binary relationship:

\[ \text{People} \rightarrow \text{Football Teams} \]

- A relationship may involve three entities, ternary relationship:

\[ \text{People} \rightarrow \text{Football Teams} \rightarrow \text{Date} \]

RELATIONSHIPS (Cont…)

- A relationship may involve \( n \) entities, \( N \)-ary relationship

The double lines specify total participation of entities in People entity set in the “fans” relationship set.

RELATIONSHIPS (Cont…)

- Example: A library database contains a listing of authors that have written books on various subjects (one author per book). It also contains information about libraries that carry books on various subjects.
RELATIONSHIPS (Cont…)

• Example: A library database contains a listing of authors that have written books on various subjects (one author per book). It also contains information about libraries that carry books on various subjects.

Entity sets: authors, subjects, books, libraries

• Relationship sets: wrote, carry, indexed

E-R diagram:

RELATIONSHIPS (Cont…)

• Example: A library database contains a listing of authors that have written books on various subjects (one author per book). It also contains information about libraries that carry books on various subjects.

Entity sets: authors, subjects, books, libraries

Relationship sets: wrote, carry, indexed

E-R diagram:

BINARY RELATIONSHIP

A binary relationship between entity set A and B might be:

• 1:1 Women marrying Men (function)

women –—* men

• N:1 Children having mothers (function)

children –—* mothers

• 1:N Mothers having children (inverse function)

mothers –—* children

• M:N Students enrolled in a class

students –—* classes

KEY (Cont…)

• A foreign key is a set of one or more attributes of a strong entity set that are employed to construct the discriminator 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 it is existence-dependent.

• Relationship sets also have primary keys. Assume R is a relationship set involving entity sets E₁, E₂,..., Eₙ. Let primary-key(Eᵢ) denote the primary key for entity set Eᵢ. Assume primary-key(Eᵢ) is unique for 1 ≤ i ≤ n. If R has no attributes then its superkey is:

\[ \text{primary-key}(E₁) \cup \text{primary-key}(E₂) \cup ... \cup \text{primary-key}(Eₙ) \]

• This is a primary key if the mapping constraint is many-to-many.

• If the mapping constraint is many-to-one from Eᵢ to Eⱼ then the primary key of R is primary key of Eᵢ.

KEY

• Entities and relationships are distinguishable using various keys

• A key is a combination of one or more attributes, e.g., social-security number, combination of name and social-security number.

• A superkey is a key defined either for an entity set or relationship set that uniquely identifies an entity, e.g., social-security number, phone number, combination of name and social-security number.

• A candidate key is a minimal superkey that uniquely identifies either an entity or a relationship, e.g., social-security number, phone number.

• A primary key is a candidate key that is chosen by the database designer to identify the entities of an entity set.

EXAMPLE

• Employees of a large company, e.g., IBM, where an employee reports to a manager. The manager is also an employee who reports to another manager. This chain of command continues to the very top where the CEO is the only employee who is not reporting to a manager. Draw the ER diagram for this example.
GENERALIZATION AND SPECIALIZATION

- **Generalization** is the result of computing the union of two or more entity sets to produce a higher-level entity set. It represents the containment relationship that exists between the higher-level entity set and one or more lower-level entity sets.

- **Specialization** constructs the lower-level entity sets that are a subset of a higher-level entity set.

  \[ \text{student} \quad \text{graduate} \]

  \[ \text{Undergrad} \quad \text{graduate} \]

  \[ \text{Salaried} \quad \text{Part-time} \]

  \[ \text{Emp} \quad \text{name} \]

  \[ \text{Part-time} \quad \text{Salaried} \]

  \[ \text{Undergrad} \quad \text{graduate} \]

  \[ \text{Specialization Generalization} \]

- **Undergrad** and **graduate** are termed **subclasses** of the **superclass** **student**.

- This is a **superclass/subclass** or **simple class/subclass** relationship.

- A member of a subclass MUST be a member of the superclass.

- An alternative notation is the **Union symbol**.

- The circle with **d** specifies that the specializations are disjoint. A member of **Undergrad** entity set may NOT be a member of the **graduate** entity set.

- A design may require all members of an entity-set to be specialized. For example, an employee MUST be a member of either a Salaried or Part-time. Use double lines to dictate this constraint.

- One may allow the specialized entity sets to overlap. For example, an entity might be both a Salaried and Part-time. "o" stands for Overlap when specializing.

BASIC CONCEPTS

- Relational data model consists of a collection of tables, each with a unique name. A table is termed a relation.

- A row (also termed a tuple or a record) in table R represents a relationship among a set of values. Each value corresponds to an attribute or a field. A column of a table represents an attribute. Given a relation with attributes \((A_1, D_1, A_2, D_2, \ldots, A_n, D_n)\), the domain of this relation is \(D_1 \times D_2 \times \ldots \times D_n\). However, an instance of a relation is a subset of this domain (recall "value & a variable"), e.g., CSCI485 (sid:integer, name:string, gpa:float)

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Adams</td>
<td>3.8</td>
</tr>
<tr>
<td>102</td>
<td>Jones</td>
<td>2.9</td>
</tr>
<tr>
<td>103</td>
<td>Hayes</td>
<td>3.5</td>
</tr>
<tr>
<td>104</td>
<td>Smith</td>
<td>3.1</td>
</tr>
</tbody>
</table>
• A tuple variable \( t \) refers to a tuple in the relation. One may access the value of an attribute contained in a tuple variable by \([\text{attr-name}]\). e.g., \( t[\text{name}] \)

\[ t[\text{std}]=101 \quad t[\text{name}]=\text{"Hayes"} \quad t[1]=102 \]

• Database schema is the data about the data. It describes the logical design of the database.

\[ \text{student-schema} = (\text{stdid} \ , \text{name} \ , \text{gpa}) \quad \text{schema for relation student} \]

CSCI485(student-schema) CSCI485 is a relation on student-schema

• Database instance is the data in the database at a given instant in time.

• Entity and relationship sets are represented by relations

**TABLES GENERATION**

3. For a functional relationship (1:1, M:1, 1:M) from set A to set B, incorporate the primary key of B into A

<table>
<thead>
<tr>
<th>SS#</th>
<th>name</th>
<th>phone</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Author(SS#, name, phone)
Book(SS#, isbn, title)

• The number of attributes in a relation R is termed arity or degree of relation R.

• The number of tuples in a relation R is termed the cardinality of R.

**TABLES GENERATION (Cont...)**

Two ways to generate tables:

1. Approach 1:
   - A table for the higher level entity set
   - For each lower entity set, create a table that includes a column for each of the attributes of that entity set plus a column for each attribute of the primary key of the higher-level entity set

   Person(SS#, name)
   Student(SS#, gpa)
   Emp(SS#, salary)

2. Approach 2:
   - No table for the higher level entity set
   - One table for each lower entity set with the following columns:
     - One column for each local attribute
     - One column for each attribute of the higher-level set

   Student(SS#, name, gpa)
   Emp(SS#, name, salary)

**TABLES GENERATION (Cont...)**

• Transforming an entity set to a table (relation):
  1. Strong entity set with attributes \( a_1, a_2, \ldots, a_n \) represent it as a table with \( n \) unique columns (one column per attribute).
  2. Weak entity set with attributes \( a_1, a_2, \ldots, a_n \) and a foreign key \( b_1, b_2, \ldots, b_m \) represent it as a table with \( n+m \) columns, one for each of \( a_1, a_2, \ldots, a_n \) \( \cup \) \( b_1, b_2, \ldots, b_m \)

*Transforming relationship sets to tables:
  1. Relationship set involving entity sets \( E_1, E_2, \ldots, E_n \) each entity having primary key \( \text{prim-key}(E_i) \), represent it as a relation with each attribute corresponding to \( \text{prim-key}(E_i) \)
  2. if the relationship has attributes \( a_1, a_2, \ldots, a_n \) then the relation consists of:

\[ \{a_1, a_2, \ldots, a_n\} \cup \text{prim-key}(E_i) \]

**TABLES GENERATION (Cont...)**

• For specialization and generalization:

**TABLES GENERATION (Cont...)**

For specialization and generalization:

```plaintext
Person

SS# name

Student
Emp

isin

Student
Emp
```