Entity-Relationship Model

Semantic Modeling

- Data models have limited understanding of the semantics of data
- Any sophisticated interpretation of data is left to human user
- Example
  - `part_weights` and `shipment_quantities` are *semantically* different
  - An attempt to join two relations on `part_weights` and `shipment_quantities` is wrong
- Relational model can only capture some semantic aspects, such as domains
- Semantic models can be useful as an aid to systematic database design, even in the absence of direct system support for these ideas.
  - Semantic aspect derives from the attempt to represent the *meaning* of data

Overall Approach of Entity-Relationship Model

- Based on a perception of data in the real world
- Developed to facilitate database design by allowing the specification of an *enterprise schema* to represent the overall logical structure of a database
  - Useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema
- Identify a set of semantic concepts in the universe of discourse
  - Identify *objects* of interest (entities)
  - Group objects of same kind into *types*
  - Identify salient *properties* of the objects
  - Establish *relationships* between objects
- Devise a set of *symbolic* objects to represent semantic concepts
- Devise a set of formal *integrity rules* to go along with the symbolic objects
- Develop a set of formal *operators* for manipulating the symbolic objects

Entities and Entity Sets

- Entity
  - An object that can be uniquely identified

  - May be concrete or abstract
  - *Weak Entity*
    * Existence-dependent on another entity
    * Cannot exist if the other entity does not exist
    * Example – Dependents of an employee in a corporation

- Entity set
  - Set of entities of the same type
  - Conceptually similar to the *type* in programming languages
  - Variable in programming language is equivalent to entity in ER model
  - An entity may belong to more than one entity set

- Representation of entity
  - Attributes of an entity
  - Domain of an attribute – possible values for the attribute
  - Attribute maps from an entity set into a domain
  - Entity represented by a set of (attribute, value) pairs

- Subtype
  - Entity can be of several types simultaneously
  - All programmers are employees
  - `programmer` is a *subtype* of the *supertype* `employee`
  - `application-programmer` and `system-programmer` are subtypes of the supertype `programmer`
  - Type hierarchies
    * *Generalization hierarchies*
      · `employee` is a generalization of `programmer`
    * *Specialization hierarchies*
      · `programmer` is a specialization of `employee`
    * *ISA hierarchies*
      · every `programmer` "is a" employee

<u>Attribute</u>

- Essential property of an entity
- All entity of a given type have several common attributes
  - All employees have `employee-number`

- `employee-name` vs `employee-department`
- Each attribute draws its values from a corresponding *value set* (domain)
- Attributes can be
  - *Simple* or *Composite*
    * Composite attribute `employee-name` may be made up of simple attributes `first-name`, `middle-initial`, and `last-name`
  - *Key*
    * Unique within a context – employee number
  - *Single-valued* or *Multi-valued*
  - *Missing* – "unknown" or "not applicable"
  - *Base* or *Derived*
- Highly dependent on the structure of enterprise being modeled and the associated semantics

Relationships and Relationship Sets

- Association between several entities
  - Relationship `employee-dept` between `employee` and `department`
- Relationship set
  - a set of relationships of the same type
  - mathematical relation on $n \geq 2$ entity sets
- Example in book – Figures 2.1 and 2.2
- *Participant*
  - Entities involved in a relationship
- *Degree* of relationship
  - Number of participants in the relationship
- *Role*
  - Function of an entity in a relationship
  - Implicit
  - Example – *works-for* specified by `(manager, worker)`
- *Descriptive attributes* of a relationship
  - `date` a `customer` accessed an `account`

- *Total Relationship*
  - Every instance of participant entity E appears in at least one instance of relationship R
  - Every `employee` must belong to a `department`
- *Partial Relationship*
  - Not total
  - An `employee` may or may not participate in a `project`

Mapping Constraints

- *Mapping cardinalities*
  - Number of entities to which another entity can be associated via a relationship set
  - Most useful in describing binary relationship sets
  - *One-to-one*
    - Relation `phone-number` in a database of single persons
  - *One-to-many*
    - Department to employee
  - *Many-to-one*
    - Dependents to an employee
  - *Many-to-Many*
    - Several employees involved in several projects
- *Subordinate Entity* – same as weak entity
- *Dominant Entity* – has subordinate entities

Keys

- Mechanism to distinguish between entities and relationships
- *Superkey*
  - Set of one or more attributes to uniquely identify an entity in an entity set
  - Every superset of a superkey is a superkey
- *Candidate key*
  - Minimal superkey such that no subset is a superkey
- *Primary key*
  - Candidate key chosen by the DB designer

- *Weak entity set*
  - Attributes not sufficient to form a primary key
  - Contrast with *strong entity set*
  - *Discriminator* of a weak entity set
- Primary key of a relationship
  - Dependent on the mapping cardinality, and structure of attributes associated with the relationship

Entity-Relationship Diagram

- Technique to represent the logical structure of a database in a pictorial manner
- Entity sets
  - Represented by a rectangle
  - Rectangle labeled with the name of the entity
  - Double the rectangle to show a weak entity
- Attributes
  - Represented by ellipsis
  - Ellipsis labeled with the attribute name
  - Ellipsis attached to the relevant entity by a straight line
  - Ellipsis is dotted if the attribute is derived
  - Ellipsis is doubled if the attribute is multivalued
  - Composite attribute is depicted by further ellipses connected to the main ellipsis by using a straight line
  - Key attributes are underlined
  - No values are to be shown
- Relationship sets
  - Represented by a diamond
  - Diamond labeled with the name of relationship
  - Double the diamond if the relationship is between a weak and a strong entity
  - Diamonds connected to two entities by using straight lines
  - Lines are doubled if the relationship is total
- Lines – Links from attributes to entity sets and entity sets to relationship sets
- Lines can be directed with arrowheads to depict the type of relationship (like one-to-one)

- Directed line – depicts a one-to-one or many-to-one relationship
- Undirected line – depicts a one-to-many or many-to-many relationship

- Lines can be depicted to indicate roles (like manager)

Reducing E-R Diagrams to Tables

- E-R diagram provides an abstract database design
- E-R diagram may leave a number of details unspecified (like integrity details)
- Representation of strong entity sets
  - Strong entity set E
  - Attributes $a_1, a_2, \ldots, a_n$
  - Represent E by a table named E with n distinct columns labeled with the attributes
  - Each row in the table corresponds to one distinct entity
  - Set of all possible entities is given by the *cartesian product* of all the attributes, or

$$D_1 \times D_2 \times \cdots \times D_n$$

  where D_i represents all the possible values for attribute a_i

- Representation of weak entity sets
  - Weak entity A with attributes $a_1, a_2, \ldots, a_r$
  - Existence dependent on entity B with primary key as $b_i, b_2, \ldots, b_s$
  - Table for entity set A contains one column for each attribute in the set

$$(a_1, a_2, \ldots, a_r) \cup (b_1, b_2, \ldots, b_s)$$

- Representation of relationship sets
  - Relationship R with entity sets $E_1, E_2, \ldots, E_m$
  - n attributes in the relationship
  - Relationship represented by a table R with n distinct columns corresponding to `attribute`(R)
  - Table to show a relationship between a weak entity and a strong entity is redundant

Generalization

- Expresses commonality between different entities
- Checking account vs savings accounts
- *Higher level entity set*

- Lower level entity set
- Depicted through a triangle component labeled ISA
- Account
  - Account number
  - Balance
  - ISA
    - savings-account
      - interest-rate
    - checking-account
      - overdraft-amount
- Emphasizes the similarities between lower level entity types and hides their differences
- Attribute inheritance
  - Attributes of higher-level entity sets (supertypes) are inherited by lower-level entity sets (subtypes)
- Two methods to create tables
  - Method 1:
    - Create table for supertypes as already done
    - For each subtype, create a table with the attributes of subtype plus the primary key of the supertype
  - Method 2:
    - Do not create a table for supertype
    - Create a table for each subtype that also includes a column for each attribute of the supertype

<u>Aggregation</u>

- Limitation of E-R model
  - Not possible to express relationships between relationships
  - Figure 2.20 in the book
  - Employee *works for* project
  - Employee *uses* machinery *to work for* project
- Aggregation allows us to treat relationships as higher-level entities
- Construct entity `work` using relationship set `work` and entity sets `employee` and `project`

Design of an E-R Database Scheme

- Options for the database designer
  - ternary vs binary relationships
  - entity set vs relationship set to express a real-world concept
  - use of an entity or attribute set
  - use of strong or weak entity sets
  - generalization
  - aggregation
- Mapping cardinalities
  - `customer-account-branch`
  - `customer-branch-account`
- Use of entity sets or relationship sets
  - representing `account` as relationship instead of entity
  - problem if several `customers` jointly hold an `account`
- Use of extended E-R features
  - Double rectangles and diamonds
  - Modular approach