Converting EER diagrams to relations and relationships

EER to Tables: Overview/Agenda

- Relational Integrity Constraints – Domain, Entity, Referential
- Transforming ER/EER diagrams to relational Schema/tables
  - Regular Entity
  - Composite Attribute
  - Multi-valued Attribute
  - Weak Entity
  - Binary Relationships
  - Supertype / Subtype

Properties of relations (revisited)

- Every relation must have a distinct name
- Each cell of the relation is atomic
  - A particular tuple can have only one value for an attribute
    - An attribute may sometimes have a NULL value, however
      - If there is no applicable value, or if the value is unknown
- Each attribute (column) in a table must have a distinct name
- The values of any attribute must be from the same domain and thus must have the same data format
- There should not be any duplicate tuples
- Order of tuples and/or attributes has no significance
Relational Data Integrity

- Two main types of database integrity constraints
  - Entity Integrity
  - Referential (or Reference) Integrity

Relational database keys

- **Superkey**: An attribute (or combination of attributes) that uniquely identifies each row in a table
- **Candidate key**: A minimal (irreducible) superkey
  - No subset of attributes in a candidate key can be a superkey
- **Primary key**: A candidate key selected to be used as the primary row identifier
  - Cannot contain NULL entries
- **Foreign key**: An attribute (or combination of attributes) in one table whose values must either match the primary key in another table, or be NULL
- **Secondary key**: Attribute (or attributes) used for data retrieval

Choosing a Primary Key

STUDENT (ST_SSN, ST_ID, ST_LNAME, ST_FNAME, ST_PHONE, ST_EMAIL)

**Superkeys**:
- ST_ID
- ST_SSN
- ST_ID, ST_LNAME
- ST_FNAME, ST_LNAME, ST_PHONE, ST_EMAIL
- etc.

**Candidate keys**:
- ST_ID
- ST_SSN (if a superkey)
- ST_EMAIL
- ST_FNAME, ST_LNAME, ST_PHONE (if a superkey)

The Primary Key for STUDENT is then just a choice from the list of candidate keys
Entity Integrity

• Entity integrity implies that:
  – Every relation in a relational data model has a unique primary key
  – The data values for all attributes in the primary key are not NULL

• Enforcing this rule ensures that every row has a unique identity and thus allows foreign key values to properly reference primary key values

• EX: Customers must be uniquely identified by their ID number

Referential Integrity

• Referential integrity implies that:
  – Every non-NULL foreign key references an existing primary key value

• Enforcing this rule makes it impossible to have an invalid foreign key value, or to delete a row whose primary key has matching foreign key values in another table

Some observations

• ER diagrams enable us to understand not only the data, but also the processes

• They provide a framework for capturing data, and the relationship(s) between the data
  – ER diagrams capture the “context”…but individual relations do not
  – The DBMS does not distinguish between one relation and another – it is up to the data modeler to understand the context
Observations (cont.)

• ER diagrams can be complex (consider for example multi-valued attributes, associative entities, strong/weak entities, or supertype/subtype relationships) – however, relations cannot
  – We have to suitably break down the complexity
  – Relations present one consistent method of representing the data — *everything is in the form of tables*
  • …whether we are representing entities or relationships or supertype/subtype or associative entities…

The end result is a set of related tables, no matter how complex the ER diagram is

![Diagram showing tables and foreign keys]

You define the tables with their primary keys, and if required, the foreign keys that reference primary keys of other tables.

Standard notation is to have a foreign key point to its corresponding primary key.

Four things that we need to be able to build into a set of relations

• Entities
  – With simple attributes
  – With composite attributes
  – With multi-valued attributes
  – Weak entities
• Relationships
  – One-to-many (1:M)
  – One-to-one (1:1)
  – Many-to-many (M:N)
• Associative entities
• Supertypes/subtypes
From Entities to Relations

Mapping Regular Entities to Relations:
1. **Simple attributes**: ER attributes map directly onto the relation
2. **Composite attributes**: Use only their simple, component attributes
3. **Multi-valued attribute**: Becomes a separate relation with a foreign key taken from the superior entity

Mapping a regular entity
(a) CUSTOMER entity type with simple attributes
(b) CUSTOMER relation

Mapping a composite attribute
(a) CUSTOMER entity type with composite attribute
(b) CUSTOMER relation with address detail
Mapping a multi-valued attribute

(a) EMPLOYEE Entity Type with multi-valued attribute

(b) Multi-valued attribute becomes a separate relation with foreign key

Mapping Weak Entities

• A Weak entity becomes a relation with a foreign key that is adopted from its associated Strong entity
• The primary key for the Weak entity relation is composed of:
  – Partial identifier of Weak entity
  – Primary key of identifying relation (Strong entity)

  = Composite primary key (multiple attributes)

Mapping Weak Entities: the EMPLOYEE Example
The resulting relations

- **EMPLOYEE**
  - Employee_ID
  - Employee_Name

- **DEPENDENT**
  - First_Name
  - Middle_initial
  - Last_Name
  - Employee_ID
  - Date_of_Birth
  - Gender

  Foreign key

  Composite primary key

Relationships: Mapping binary relationships

- **One-to-Many**: Primary key on the "One" side becomes a foreign key on the "Many" side
- **One-to-One**: Primary key on the mandatory side becomes a foreign key on the optional side
- **Many-to-Many**: Create a *new relation* with the primary keys of the two entities as its primary key

One-to-Many (1:M) Binary Relationship

- Note the mandatory "One"
Mapping the 1:M relationship

One-to-One (1:1) Binary Relationship

Many-to-Many (M:N) Binary Relationship
Many-to-Many (M:N) Binary Relationship

Step 2 of 2:

Mapping supertype/subtype relationships

- Define one relation for the supertype and one for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype-specific attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table

Supertype/subtype example