Chapter 5
Database Design

Elements of Database Design

The Database Design Process
- Create tables and columns from entities and attributes
- Select primary keys
- Represent relationships
- Specify constraints
- Re-examine normalization criteria

Selecting the Primary Key
- An ideal primary key is short, numeric, and seldom changing
- If there are more than one candidate keys (alternate identifiers), they should be evaluated and the best one chosen as the table’s primary key
- If the entity has no identifier, an attribute needs to be selected as the identifier
- In some situations, a surrogate key should be defined

Surrogate Keys
- A surrogate key is a unique, DBMS-supplied identifier used as the primary key of a relation
- The values of a surrogate key have no meaning to the users and are normally hidden on forms and reports
- DBMS does not allow the value of a surrogate key to be changed
- Disadvantages:
  - Foreign keys that are based on surrogate keys have no meaning to the users
  - When data shared among different databases contain the same ID, merging those tables might yield unexpected results
Example: Surrogate Keys

Representing Relationships
- Relationships are expressed by placing the primary key of one table into a second table
- The new column in the second table is referred to as a **foreign key**
- Three principles of relationship representation
  - Preservation of referential integrity constraints
  - Specification of referential integrity actions
  - Representation of minimum cardinality

Rules for Referential Integrity Constraints

Specifying Referential Integrity Actions
- If default referential integrity constraint is too strong, overriding the default referential integrity enforcement could be defined during database design
- The policy will be programmed into triggers during implementation
- Two referential integrity overrides
  - **Cascading updates** automatically change the value of the foreign key in all related child rows to the new value
  - **Cascading deletions** automatically delete all related child rows

Enforcing Minimum Cardinality
- If the minimum cardinality on the child is one, at least one child row must be connected to the parent
- A required parent can be specified by making the foreign key value not null
- A required child can be represented by creating update and delete referential integrity actions on the child and insert referential integrity actions on the parent
- Such referential integrity actions must be declared during database design and trigger codes must be written during implementation

Representing ID-Dependent Relationships
- To represent ID-dependent relationships, primary key of the parent relation is added to the child relation
- The new foreign key attribute becomes part of the child’s composite primary key
- Referential integrity actions should be carefully determined
  - For cascading updates, data values are updated to keep child rows consistent with parent rows
  - If the entity represents multi-value attributes, cascading deletions are inappropriate
  - Check user requirements when designing more complex situations
Example: ID-Dependent Relationship

If the parent in an ID-dependent relationship has a surrogate key as its primary key, but the child has a data key, use the parent’s surrogate key as a primary key.

A mixture of a surrogate key with a data key does not create the best design as the composite key will have no meaning to the users.

Therefore, whenever any parent of an ID-dependent relationship has a surrogate key, the child should have a surrogate key as well.

By using surrogate keys in the child table, the relationship type has changed to 1:N non-identifying relationship.

Example: Cascading Deletion

Representing 1:1 and 1:N Relationships

- IDEF1X refers to 1:1 and 1:N as non-identifying connection relationships.
- General rule: the key of a parent table is always placed into the child.
  - For 1:1 relationship, either entity could be considered the parent or the child.
  - For 1:N relationship, the parent entity is always the entity on the one side.

Example: 1:1 Relationship

Representing Relationship Using Surrogate Keys

- If the parent in an ID-dependent relationship has a surrogate key as its primary key, but the child has a data key, use the parent’s surrogate key as a primary key.
- A mixture of a surrogate key with a data key does not create the best design as the composite key will have no meaning to the users.
- Therefore, whenever any parent of an ID-dependent relationship has a surrogate key, the child should have a surrogate key as well.
- By using surrogate keys in the child table, the relationship type has changed to 1:N non-identifying relationship.
Representing N:M Relationships

- IDEF1X refers to N:M relationships as non-specific relationships.
- N:M relationships need to be converted into two ID-dependent relationships by defining an intersection table.
- Two referential integrity constraints will be created:
  - The minimum cardinality from the child to the parent is always one.
  - The minimum cardinality from the parent to the intersection table depends on the system requirements.

N:M Relationships Suggesting Missing Entities

- According to IDEF1X, N:M relationship suggests a possible missing entity:
  - If there is a missing entity, that entity will be ID-dependent on both of its parents.
  - If there is no missing entity, create the connecting entity with no non-key attributes.
- This approach is similar to the representation of N:M relationship in extended E-R model using intersection table.
Representing Subtype Relationships

- Called subtypes in the extended E-R model and categories in the IDEF1X model
- Primary key of the supertype (or generic) entity is placed into the subtype (or category entity)
- Category entities in IDEF1X are mutually exclusive in the categories
  - For complete categories, the generic entity will have exactly one category entity in that cluster
  - These constraints are enforced by properly specifying referential integrity actions

Representing Weak Entities

- Weak entities logically depend on the existence of another entity in the database
- Representing these entities are the same as modeling 1:1 or 1:N relationships
- Referential integrity actions need to be specified to ensure that
  - When the parent is deleted, the weak entity is deleted as well
  - New weak entities have a parent with which to connect
Example: Nested ID-Dependent Relationships

Example: University System

Example: University System

Example: 1:1 Recursive Relationships

Example: 1:N Recursive Relationships

Representing Recursive Relationships

- A recursive relationship is a relationship among entities of the same class
- For 1:1 and 1:N recursive relationships, add a foreign key to the relation that represents the entity
- For N:M recursive relationships, add a new intersection table that represents the N:M relationship
Example: M:N Recursive Relationships

Representing Ternary and Higher-Order Relationships

- Ternary and higher-order relationships can be treated as combinations of binary relationships
- There are three types of binary constraints: MUST, MUST NOT, and MUST COVER
  - MUST NOT constraint: the binary relationship indicates combinations that are not allowed to occur in the ternary relationship
  - MUST COVER constraint: the binary relationship indicates all combinations that must appear in the ternary relationship
- Because none of these constraints can be represented in the relational design, they must be documented as business rules and enforced in application programs or triggers

Null values

- A null value is an attribute value that has not been supplied
- Null values are ambiguous as they can mean
  - The value is unknown
  - The value is inappropriate
  - The value is known to be blank
- Inappropriate nulls can be avoided by
  - Defining subtype or category entities
  - Forcing attribute values through the use of not null
  - Supplying initial values
- Ignore nulls if the ambiguity is not a problem to the users