

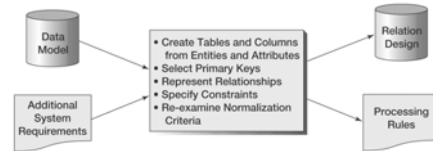
## Chapter 5 Database Design



**DATABASE PROCESSING**  
Fundamentals, Design,  
and Implementation, 9/e

## Elements of Database Design

Figure 5.1 Elements of Database Design



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 5/2

## The Database Design Process

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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 5/3

## Transforming an Entity to a Table

Figure 5.2 Transforming an Entity to a Table  
(a) BUILDING Entity; (b) BUILDING Table and  
(c) Alternative Table Diagram



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 5/4

## 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 5/5

## Surrogate Keys

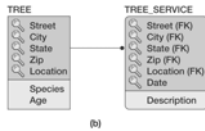
- A surrogate key is a unique, DBMS-supplied identifier used as the primary key of a relation
- The values of a surrogate key have no meaning to the users and are normally hidden on forms and reports
- DBMS does not allow the value of a surrogate key to be changed
- Disadvantages:
  - Foreign keys that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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e  
by David M. Kroenke

Chapter 5/6

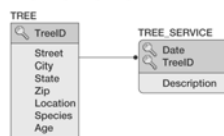
## Example: Surrogate Keys

Figure 5.4b Need for Surrogate Keys — Migration of Large Primary Key to Second Table



(b)

Figure 5.4 Need for Surrogate Keys — Design Using Surrogate Keys



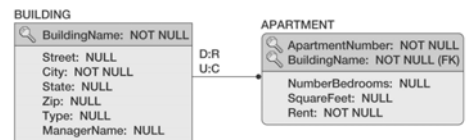
(c)

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/7

## Example: Surrogate Keys

Figure 5.8c Representation of ID-dependent Relationships — Specifying Referential Integrity Actions



(c)

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/8

## 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/9

## Rules for Referential Integrity Constraints

Most relational DBMS products enforce this when a foreign key is used (Access has a check box labeled "Enforce Referential Integrity").

Figure 5.6 Default Rules for Enforcing Referential Integrity

Action on Parent	Insert new row	Insert always OK.
Update primary key	Disallow update if parent row has child rows.	
Delete row	Disallow deletion if parent row has child rows.	
Action on Child	Insert new row	Disallow insert if foreign key in new row does not match a primary key value in the parent table.
Update foreign key	Disallow update if updated foreign key does not match a primary key value in the parent table.	
Delete existing row	Delete always OK.	

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/10

## 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, in some cases, need to 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/11

## 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 must be represented by creating update and delete referential integrity actions on the child and insert referential integrity actions on the parent (a feature not supplied by most DBMS)
- Such referential integrity actions must be declared during database design and *trigger codes* must be written during implementation

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/12

## Representing ID-Dependent Relationships

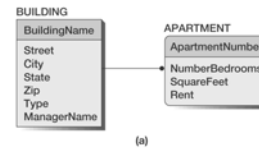
- To represent ID-dependent relationships, primary key of the parent relation is added to the child relation
- The new foreign key attribute becomes part of the child's composite primary key
- Referential integrity actions should be carefully determined
  - For cascading updates, data values are updated to keep child rows consistent with parent rows
  - If the entity represents *multi-value dependencies*, cascading deletions are appropriate
  - Check user requirements when designing more complex situation

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/13

## Example: ID-Dependent Relationship

Figure 5.8a Representation of ID-dependent Relationships — ID-dependent Relationship Example



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/14

## Example: ID-Dependent Relationship

Figure 5.8b Representation of ID-dependent Relationships — Relationship Design for Example in (a)

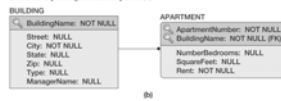
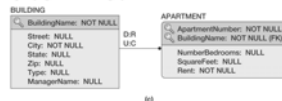


Figure 5.8c Representation of ID-dependent Relationships — Specifying Referential Integrity Actions

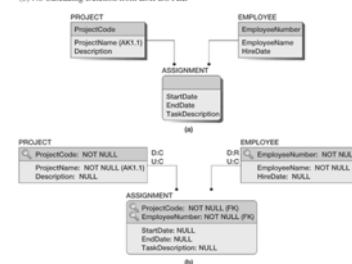


Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/15

## Example: Cascading Deletion

Figure 5.10 Mixed Cascading Behavior (a) Example with Two Relationships and (b) No Cascading Deletion from EMPLOYEE



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/16

## 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 in the child
- 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/17

## Representing 1:1 and 1:N "HAS-A" Relationships

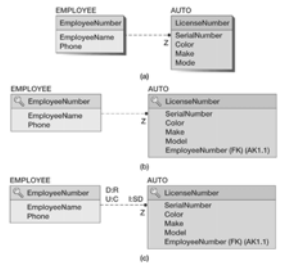
- IDEF1X refers to 1:1 and 1:N as Non-identifying connection relationships
- General rule: the key of a parent table is always placed into the child
  - For 1:1 relationship, either entity could be considered the parent or the child
  - For 1:N relationship, the parent entity is always the entity on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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/18

## Example: 1:1 Relationship

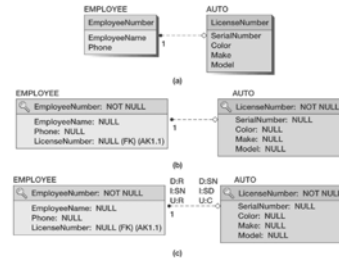
Figure 5.12 1:1 with EMPLOYEE Considered as Parent



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/19

## Example: 1:1 Relationship

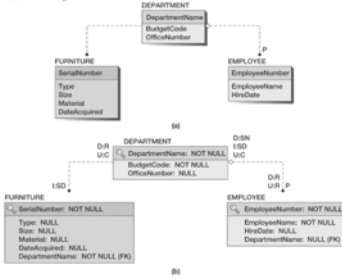
Figure 5.13 1:1 with AUTO Considered as Parent



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/20

## Example: 1:N Relationship

Figure 5.14 Two Example 1:N Relationships



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/21

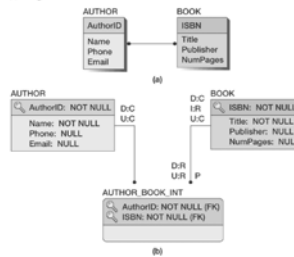
## Representing N:M Relationships

- 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/22

## Example: N:M Relationship

Figure 5.16 N:M Relationship Example



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/23

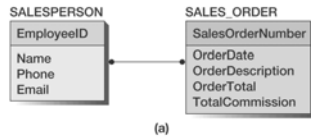
## N:M Relationships Suggesting Missing Entities

- 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/24

## Example: Missing Entity

**Figure 5.17a** N:M Relationship with Missing Entity — Non-Specific Relationship

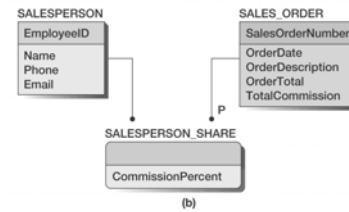


Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/25

## Example: Missing Entity

**Figure 5.17b** N:M Relationship with Missing Entity — Model with Missing Entity

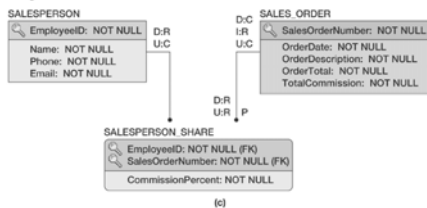


Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/26

## Example: Missing Entity

**Figure 5.17c** N:M Relationship with Missing Entity — Design without Intersection Table



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/27

## Representing Subtype Relationships

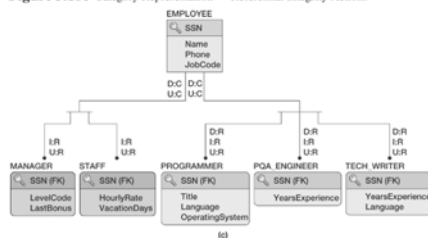
- Called subtypes in the extended E-R model and categories in the IDEF1X model
- Primary key of the supertype (or generic) entity is the primary key in the subtype (or category entity)
- Category entities in IDEF1X are mutually exclusive in the categories
  - For complete categories, the generic entity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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/28

## Example: Subtype Relationship

**Figure 5.18c** Category Representation — Referential Integrity Actions



Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/29

## Representing Weak Entities

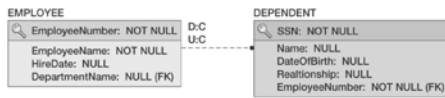
- Weak entities logically depend on the existence of another entity in the database
- Representing these entities are the same as modeling 1:1 or 1:N relationships
- Referential integrity actions need to be specified to ensure that
  - When the parent is deleted, the weak entity is deleted as well
 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke

Chapter 5/30

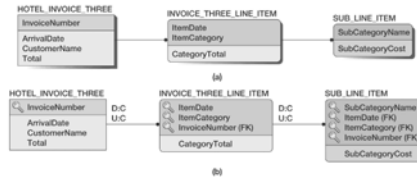
## Example: Weak, Non ID-Dependent Relationships

Figure 5.19 Weak but Not ID-Dependent Entity Table Design



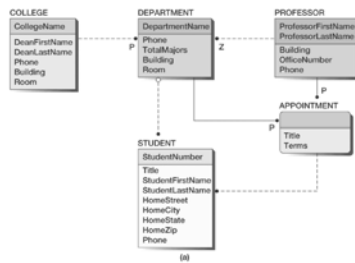
## Example: Nested ID-Dependent Relationships

Figure 5.20 Design with Nested ID-Dependent Entities (a) Example Entities and (b) Table Design



## Example: University System

Figure 5.22a Highline University Example — Highline University Data Model



## Example: University System

Figure 5.22b Highline University Example — Database Design for Highline University

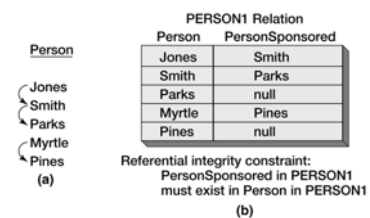


## Representing Recursive Relationships

- 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: 1:1 Recursive Relationships

Figure 5.24a&b Example of a 1:1 Recursive Relationship (a) Sample Data for 1:1 Recursive Relationship; (b) First Alternative for Representing a 1:1 Recursive Relationship



## Example: 1:N Recursive Relationships

**Figure 5.25** Example of a 1:N Recursive Relationship (a) Sample Data for the REFERRED\_BY Relationship and (b) Representing a 1:N Recursive Relationship by Means of a Relation

**(a)**

Customer Number	Referred These Customers
100	200, 400
300	500
400	600, 700

**(b)**

CUSTOMER Relation	CustomerNumber	CustomerData	ReferredBy
	100	...	null
	200	...	100
	300	...	null
	400	...	100
	500	...	300
	600	...	400
	700	...	400

Referential integrity constraint:  
ReferredBy in CUSTOMER must exist in CustomerNumber in CUSTOMER

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/37

## Example: M:N Recursive Relationships

**Figure 5.26a** Example of an M:N Recursive Relationship (a) Sample Data for the TREATED-BY Relationship

**(a)**

Provider Receiver

Jones Smith  
Parks Smith  
Smith Abernathy  
Abernathy Jones  
Franklin Franklin

**Figure 5.26b** Example of an M:N Recursive Relationship — Representing an M:N Recursive Relationship by Means of Relations

DOCTOR relation	Name	Other Attributes
	Jones	...
	Parks	...
	Smith	...
	Abernathy	...
	O'Leary	...
	Franklin	...

TREATMENT-INTERSECTION relation	Physician	Patient
	Jones	Smith
	Parks	Smith
	Smith	Abernathy
	Abernathy	Jones
	Parks	Franklin
	Franklin	Abernathy
	Jones	Abernathy

Referential integrity constraints:  
Physician in TREATMENT-INTERSECTION must exist in Name in DOCTOR  
Patient in TREATMENT-INTERSECTION must exist in Name in DOCTOR

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/38

## Representing Ternary and Higher-Order Relationships

(All are restrictions on Many-Many 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


Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/39

## 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

Copyright © 2004 Database Processing: Fundamentals, Design, and Implementation, 9/e by David M. Kroenke Chapter 5/40

## Chapter 5 Database Design



**DATABASE PROCESSING**  
Fundamentals, Design,  
and Implementation, 9/e