Chapter 7 ER Model

Using High-Level Conceptual Data Models for Database Design

A Sample Database Application

Entity Types, Entity Sets, Attributes, and Keys

Relationship Types, Relationship Sets, Roles, and Structural Constraints Weak Entity Types

Refining the ER Design for the COMPANY Database

ER Diagrams, Naming Conventions, and Design Issues

Example of Other Notation: UML Class Diagrams

Relationship Types of Degree Higher than Two

Data Modeling Using the  
Entity-Relationship (ER) Model

**Entity-Relationship (ER) model**

Popular high-level conceptual data model **ER diagrams**

Diagrammatic notation associated with the ER model

**Unified Modeling Language (UML)**

Using High-Level Conceptual Data Models for Database Design

**Requirements collection and analysis**

Database designers interview prospective database users to understand and document data requirements

Result: **data requirements**

**Functional requirements** of the application

Using High-Level Conceptual Data Models (cont’d.)

**Conceptual schema**

Conceptual design

Description of data requirements

Includes detailed descriptions of the entity types, relationships, and constraints

Transformed from high-level data model into implementation data model .)

**Logical design** or **data model mapping**

Result is a database schema in implementation data model of DBMS **Physical design phase**

Internal storage structures, file organizations, indexes, access paths, and physical design parameters for the database files specified

A Sample Database Application

COMPANY

Employees, departments, and projects

Company is organized into departments

Department controls a number of projects

Employee: store each employee’s name, Social Security number, address, salary, sex (gender), and birth date

Keep track of the dependents of each employee

Entity Types, Entity Sets, Attributes, and Keys

ER model describes data as:

Entities Relationships Attributes Entities and Attributes

**Entity** Thing in real world with independent existence

**Attributes** Particular properties that describe entity

Types of attributes:

*Composite* versus *simple* (atomic) *attributes*

**Single-valued** versus **multivalued** attributes

**Stored** versus **derived** attributes

**NULL** values

**Complex** attributes

Entity Types, Entity Sets, Keys, and Value Sets

**Entity type**

Collection (or set) of entities that have the same attributes

**Key** or **uniqueness constraint** Attributes whose values are distinct for each individual entity in entity set

**Key attribute** Uniqueness property must hold for every entity set of the entity type

**Value sets** (or **domain of values**)

Specifies set of values that may be assigned to that attribute for each individual entity

Initial Conceptual Design of the COMPANY Database

Relationship Types, Relationship Sets, Roles, and Structural Constraints

**Relationship** When an attribute of one entity type refers to another entity type Represent references as relationships not attributes

Relationship Types, Sets, and Instances

**Relationship type** *R* among *n* entity types *E*1, *E*2, ..., *En*

Defines a set of associations among entities from these entity types

**Relationship instances** *ri* Each *ri* associates n individual entities (*e*1, *e*2, ..., *en*)

Each entity *ej* in *ri* is a member of entity set *Ej*

Relationship Degree

**Degree** of a relationship type

Number of participating entity types

**Binary**, **ternary** Relationships as attributes

Think of a binary relationship type in terms of attributes

Role Names and Recursive Relationships

**Role names** and recursive relationships

Role name signifies role that a participating entity plays in each relationship instance

**Recursive** relationships

Same entity type participates more than once in a relationship type in different roles Must specify role name

Constraints on Binary Relationship Types

**Cardinality ratio** for a binary relationship

Specifies maximum number of relationship instances that entity can participate in

**Participation constraint**  Specifies whether existence of entity depends on its being related to another entity

Types: **total** and **partial** Attributes of Relationship Types

Attributes of 1:1 or 1:N relationship types can be migrated to one entity type For a 1:N relationship type

Relationship attribute can be migrated only to entity type on N-side of relationship

For M:N relationship types

Some attributes may be determined by combination of participating entities

Must be specified as relationship attributes

Weak Entity Types

Do not have key attributes of their own

Identified by being related to specific entities from another entity type **Identifying relationship** Relates a weak entity type to its owner Always has a total participation constraint

Refining the ER Design for the COMPANY Database

Change attributes that represent relationships into relationship types Determine cardinality ratio and participation constraint of each relationship type

ER Diagrams, Naming Conventions, and Design Issues

Proper Naming of Schema Constructs

Choose names that convey meanings attached to different constructs in schema

Nouns give rise to entity type names

Verbs indicate names of relationship types

Choose binary relationship names to make ER diagram readable from left to right and from top to bottom

Design Choices for ER Conceptual Design

Model concept first as an attribute

Refined into a relationship if attribute is a reference to another entity type Attribute that exists in several entity types may be elevated to an independent entity type

Can also be applied in the inverse

Alternative Notations for ER Diagrams

Specify structural constraints on relationships

Replaces cardinality ratio (1:1, 1:N, M:N) and single/double line notation for participation constraints

Associate a pair of integer numbers (min, max) with each participation of an entity type *E* in a relationship type *R*, where 0 ≤ min ≤ max and max ≥ 1

Example of Other Notation:  
UML Class Diagrams

UML methodology

Used extensively in software design

Many types of diagrams for various software design purposes

UML class diagrams

Entity in ER corresponds to an object in UML

Example of Other Notation:  
UML Class Diagrams (cont’d.)

**Class** includes three sections:

Top section gives the class name

Middle section includes the attributes;

Last section includes operations that can be applied to individual objects Example of Other Notation:  
UML Class Diagrams (cont’d.)

**Associations**: relationship types

**Relationship instances**: links Binary association

Represented as a line connecting participating classes

May optionally have a name

Link attribute

Placed in a box connected to the association’s line by a dashed line

Example of Other Notation:  
UML Class Diagrams (cont’d.)

**Multiplicities**: min..max, asterisk (\*) indicates no maximum limit on participation

Types of relationships: **association** and **aggregation**

Distinguish between **unidirectional** and **bidirectional** associations

Model weak entities using **qualified** **association**

Relationship Types of Degree  
Higher than Two

**Degree** of a relationship type

Number of participating entity types

*Binary* Relationship type of degree two

*Ternary* Relationship type of degree three

Choosing between Binary and Ternary (or Higher-Degree) Relationships

Some database design tools permit only binary relationships

Ternary relationship must be represented as a weak entity type

No partial key and three identifying relationships

Represent ternary relationship as a regular entity type

By introducing an artificial or surrogate key

Constraints on Ternary (or Higher-Degree) Relationships

Notations for specifying structural constraints on *n*-ary relationships

Should both be used if it is important to fully specify structural constraints

Summary

Basic ER model concepts of entities and their attributes

Different types of attributes

Structural constraints on relationships

ER diagrams represent E-R schemas

UML class diagrams relate to ER modeling concepts

Chapter 9 ER to Relational Mapping

ER-to-relational mapping

Relational Database Design Using ER-to-Relational Mapping

Mapping EER Model Constructs to Relations

Relational Database  
Design by ER- and EER-to-Relational Mapping

**Design a relational database schema**

Based on a conceptual schema design

Seven-step algorithm to convert the basic ER model constructs into relations

Additional steps for EER model

Relational Database Design Using ER-to-Relational Mapping

ER-to-Relational Mapping Algorithm

COMPANY database example

Assume that the mapping will create tables with simple single-valued attributes

Step 1: Mapping of Regular Entity Types

For each regular entity type, create a relation *R* that includes all the simple attributes of *E*

Called **entity relations**

Each tuple represents an entity instance

Step 2: Mapping of Weak Entity Types

For each weak entity type, create a relation *R* and include all simple attributes of the entity type as attributes of *R*

Include primary key attribute of owner as foreign key attributes of *R* Step 3: Mapping of Binary 1:1 Relationship Types

For each binary 1:1 relationship type

Identify relations that correspond to entity types participating in *R*

Possible approaches:

**Foreign key approach**

**Merged relationship approach**

**Crossreference or relationship relation approach**

ER-to-Relational Mapping Algorithm (cont’d.)

Step 4: Mapping of Binary 1:*N* Relationship Types

For each regular binary 1:*N* relationship type

Identify relation S that represents participating entity type at *N*-side of relationship type

Include primary key of other entity type as foreign key in *S*

Include simple attributes of 1:*N* relationship type as attributes of *S*

ER-to-Relational Mapping Algorithm (cont’d.)

Alternative approach

Use the **relationship relation** (cross-reference) option as in the third option for binary 1:1 relationships

ER-to-Relational Mapping Algorithm (cont’d.)

Step 5: Mapping of Binary *M*:*N* Relationship Types

For each binary *M*:*N* relationship type

Create a new relation *S*

Include primary key of participating entity types as foreign key attributes in *S*

Include any simple attributes of *M*:*N* relationship type

ER-to-Relational Mapping Algorithm (cont’d.)

Step 6: Mapping of Multivalued Attributes

For each multivalued attribute A

Create a new relation R

Primary key of *R* is the combination of *A* and *K*

If the multivalued attribute is composite, include its simple components ER-to-Relational Mapping Algorithm (cont’d.)

Step 7: Mapping of *N*-ary Relationship Types

For each *n*-ary relationship type *R*

Create a new relation *S* to represent *R*

Include primary keys of participating entity types as foreign keys

Include any simple attributes as attributes

In a relational schema relationship, types are not represented explicitly Represented by having two attributes *A* and *B*:one a primary key and the other a foreign key

Mapping EER Model Constructs to Relations

Extending ER-to-relational mapping algorithm

Mapping of Specialization or Generalization

Step 8: Options for Mapping Specialization or Generalization (see pages 294-295) **Option 8A: Multiple relations—superclass and subclasses** For any specialization (total or partial, disjoint or overlapping) **Option 8B: Multiple relations—subclass relations only** Subclasses are total Specialization has disjointedness constraint Mapping of Specialization or Generalization (cont’d.)

**Option 8C: Single relation with one type attribute**

Type or discriminating attribute indicates subclass of tuple

Subclasses are disjoint

Potential for generating many NULL values if many specific attributes exist in the subclasses

**Option 8D: Single relation with multiple type attributes**

Subclasses are overlapping

Will also work for a disjoint specialization

Mapping of Shared Subclasses (Multiple Inheritance)

Apply any of the options discussed in step 8 to a shared subclass

Mapping of Categories (Union Types)

Step 9: Mapping of Union Types (Categories)

Defining super classes have different keys

Specify a new key attribute

**Surrogate key**

Summary

Map conceptual schema design in the ER model to a relational database schema

Algorithm for ER-to-relational mapping

Illustrated by examples from the COMPANY database

Include additional steps in the algorithm for mapping constructs from EER model into relational model

Chapter 14 Web Database Programming using PHP

Comes installed with the UNIX operating system

DBMS

**Bottom-tier database server** PHP

**Middle-tier Web server** HTML

**Client tier**

PHP script stored in:

http://www.myserver.com/example/greeting.php

$\_POST

**Auto-global** predefined PHP variable

Array that holds all the values entered through form parameters

Arrays are dynamic

**Long text strings**

Between opening <<<\_HTML\_ and closing \_HTML\_;

PHP **variable names**

Start with $ sign

Overview of Basic Features of PHP

Illustrate features of PHP suited for creating dynamic Web pages that contain database access commands

PHP Variables, Data Types, and Programming Constructs

PHP **variable names**

Can include characters, letters, and underscore character (\_)

Main ways to express strings and text

**Single-quoted strings**

**Double-quoted strings**

**Here documents**

**Single and double quotes**

Period (.) symbol String concatenate operator

Single-quoted strings

Literal strings that contain no PHP program variables

Double-quoted strings and here documents

Values from variables need to be interpolated into string

Numeric data types

Integers and floating points

Programming language constructs

For-loops, while-loops, and conditional if-statements

Boolean expressions

Comparison operators

== (equal), != (not equal), > (greater than), >= (greater than or equal), < (less than), and <= (less than or equal)

PHP Arrays

Database query results

Two-dimensional arrays

First dimension representing rows of a table

Second dimension representing columns (attributes) within a row

Main types of arrays:

**Numeric** and **associative**

PHP Arrays (cont’d.)

Numeric array

Associates a numeric index with each element in the array

Indexes are integer numbers

Start at zero Grow incrementally

Associative array Provides pairs of (key => value) elements

Techniques for looping through arrays in PHP

Count function

Returns current number of elements in array

Define to structure a complex program and to share common sections of code

Arguments passed by value

Examples to illustrate basic PHP functions

Built-in entries

$\_SERVER auto-global built-in array variable

Provides useful information about server where the PHP interpreter is running Examples:

**$\_SERVER['SERVER\_NAME']**

**$\_SERVER['REMOTE\_ADDRESS']**

**$\_SERVER['REMOTE\_HOST']**

**$\_SERVER['PATH\_INFO']**

**$\_SERVER['QUERY\_STRING']**

**$\_SERVER['DOCUMENT\_ROOT']**

$\_POST

Provides input values submitted by the user through HTML forms specified in <INPUT> tag

Overview of PHP Database Programming

PEAR DB library

Part of PHP Extension and Application Repository (PEAR)

Provides functions for database access

Connecting to a Database

Library module DB.php must be loaded

DB library functions accessed using DB::<function\_name>

DB::connect('string')

Function for connecting to a database

Format for 'string' is: <DBMS software>://<user account>:<password>@<database server>

Connecting to a Database (cont’d.)

Query function $d->query takes an SQL command as its string argument Sends query to database server for execution

$d–>setErrorHandling(PEAR\_ERROR\_DIE)

Terminate program and print default error messages if any subsequent errors occur

Collecting Data from Forms and Inserting Records

Collect information through HTML or other types of Web forms

Create unique record identifier for each new record inserted into the database

PHP has a function $d–>nextID to create a sequence of unique values for a particular table

**Placeholders**

Specified by ? symbol

Retrieval Queries from Database Tables

$q **Query result**

$q->fetchRow() retrieve next record in query result and control loop

$d=>getAll

Holds all the records in a query result in a single variable called $allresult

Summary

PHP scripting language

Very popular for Web database programming

PHP basics for Web programming

Data types

Database commands include:

Creating tables, inserting new records, and retrieving database records

Chapter 15 Basics of Functional Dependencies and Normalization for RD, FD First/Second/Third normal form

Levels at which we can discuss *goodness* of relation schemas

Logical (or conceptual) level

Implementation (or physical storage) level

Approaches to database design:

Bottom-up or top-down

Informal Design Guidelines  
for Relation Schemas

Measures of quality

Making sure attribute semantics are clear

Reducing redundant information in tuples

Reducing NULL values in tuples

Disallowing possibility of generating spurious tuples

Imparting Clear Semantics to Attributes in Relations

Semantics of a relation

Meaning resulting from interpretation of attribute values in a tuple

Easier to explain semantics of relation

Indicates better schema design

Design relation schema so that it is easy to explain its meaning

Do not combine attributes from multiple entity types and relationship types into a single relation

Redundant Information in Tuples and Update Anomalies

Grouping attributes into relation schemas

Significant effect on storage space

Storing natural joins of base relations leads to **update anomalie**

Types of update anomalies:

Insertion, Deletion, Modification

Guideline 2 Design base relation schemas so that no update anomalies are present in the relations

If any anomalies are present:

Note them clearly

Make sure that the programs that update the database will operate correctly

NULL Values in Tuples

May group many attributes together into a “fat” relation

Can end up with many NULLs

Problems with NULLs

Wasted storage space

Problems understanding meaning

Guideline 3 Avoid placing attributes in a base relation whose values may frequently be NULL

If NULLs are unavoidable:

Make sure that they apply in exceptional cases only, not to a majority of tuples

Generation of Spurious Tuples

Relation schemas EMP\_LOCS and EMP\_PROJ1

NATURAL JOIN

Result produces many more tuples than the original set of tuples in EMP\_PROJ Called **spurious tuples**

Represent spurious information that is not valid

Guideline 4 Design relation schemas to be joined with equality conditions on attributes that are appropriately related

Guarantees that no spurious tuples are generated

Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations

Summary and Discussion of Design Guidelines

Anomalies cause redundant work to be done

Waste of storage space due to NULLs

Difficulty of performing operations and joins due to NULL values

Generation of invalid and spurious data during joins

Functional Dependencies

Formal tool for analysis of relational schemas

Enables us to detect and describe some of the above-mentioned problems in precise terms

Theory of functional dependency

Definition of Functional Dependency

Constraint between two sets of attributes from the database

Property of semantics or meaning of the attributes

**Legal relation states**

Satisfy the functional dependency constraints

Given a populated relation

Cannot determine which FDs hold and which do not

Unless meaning of and relationships among attributes known

Can state that FD does not hold if there are tuples that show violation of such an FD

Normal Forms Based on Primary Keys

Normalization process

Approaches for relational schema design

Perform a conceptual schema design using a conceptual model then map conceptual design into a set of relations

Design relations based on external knowledge derived from existing implementation of files or forms or reports

Normalization of Relations

Takes a relation schema through a series of tests

Certify whether it satisfies a certain normal form

Proceeds in a top-down fashion

**Normal form tests**

Properties that the relational schemas should have:

**Nonadditive join property**

Extremely critical **Dependency preservation property**

Desirable but sometimes sacrificed for other factors

Practical Use of Normal Forms

Normalization carried out in practice

Resulting designs are of high quality and meet the desirable properties stated previously

Pays particular attention to normalization only up to 3NF, BCNF, or at most 4NF

Do not need to normalize to the highest possible normal form

Definitions of Keys and Attributes Participating in Keys

Definition of **superkey** and **key**

**Candidate key**

If more than one key in a relation schema

One is **primary key**

Others are **secondary keys**

First Normal Form

Part of the formal definition of a relation in the basic (flat) relational model Only attribute values permitted are single **atomic (or indivisible) values** Techniques to achieve first normal form

Remove attribute and place in separate relation

Expand the key Use several atomic attributes

First Normal Form (cont’d.)

Does not allow **nested relations**

Each tuple can have a relation within it

To change to 1NF:

Remove nested relation attributes into a new relation

Propagate the primary key into it

**Unnest** relation into a set of 1NF relations

Second Normal Form

Based on concept of **full functional dependency**

Versus **partial dependency**

Second normalize into a number of 2NF relations

Nonprime attributes are associated only with part of primary key on which they are fully functionally dependent

Third Normal Form

Based on concept of transitive dependency

Problematic FD

Left-hand side is part of primary key

Left-hand side is a nonkey attribute

General Definitions of Second  
and Third Normal Forms

General Definitions of Second  
and Third Normal Forms (cont’d.)

**Prime attribute** Part of any candidate key will be considered as prime Consider partial, full functional, and transitive dependencies with respect to all candidate keys of a relation

General Definition of Second Normal Form

General Definition of Third Normal Form

Boyce-Codd Normal Form

Every relation in BCNF is also in 3NF

Relation in 3NF is not necessarily in BCNF

Difference:

Condition which allows A to be prime is absent from BCNF

Most relation schemas that are in 3NF are also in BCNF

Multivalued Dependency  
and Fourth Normal Form

Multivalued dependency (MVD)

Consequence of first normal form (1NF)

Multivalued Dependency  
and Fourth Normal Form (cont’d.)

Relations containing nontrivial MVDs

**All-key relations** **Fourth normal form (4NF)**

Violated when a relation has undesirable multivalued dependencies

Join Dependencies  
and Fifth Normal Form

**Join dependency** Multiway decomposition into fifth normal form (5NF) Very peculiar semantic constraint

Normalization into 5NF is very rarely done in practice

Join Dependencies  
and Fifth Normal Form (cont’d.)

Summary

Informal guidelines for good design

Functional dependency

Basic tool for analyzing relational schemas

Normalization:

1NF, 2NF, 3NF, BCNF, 4NF, 5NF