Informática y Comunicaciones

Chapter 7
The Entity-Relationship Model
Chapter 7: outline

7.1 SQL
7.2 The Entity-Relationship Model
Structured Query Language

- Acronym: SQL
- Pronounced as “S-Q-L” [“Ess-Que-El”]
- Originally developed by IBM as the SEQUEL language in the 1970s
- SQL:2008 is current standard.
SQL Defined

- SQL is not a programming language, but rather a data sublanguage.
- SQL is comprised of
  - A data definition language (DDL)
    - Used to define database structures
  - A data manipulation language (DML)
    - Data definition and updating
    - Data retrieval (Queries)
The SQL data definition statements include:

- **CREATE**
  - To create database objects
- **ALTER**
  - To modify the structure and/or characteristics of database objects
- **DROP**
  - To delete database objects
- **TRUNCATE**
  - To delete table data while keeping structure
SQL for Data Definition: CREATE

- Creating database tables
  - The SQL CREATE TABLE statement

```sql
CREATE TABLE EMPLOYEE(
    EmpID       Integer PRIMARY KEY,
    EmpName     Char(25) NOT NULL
);
```
SQL for Data Definition:
CREATE with CONSTRAINT I

- Creating database tables with PRIMARY KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```sql
CREATE TABLE EMPLOYEE(
    EmpID Integer NOT NULL,
    EmpName Char(25) NOT NULL
CONSTRAINT Emp_PK PRIMARY KEY(EmpID)
);
```
SQL for Data Definition:  
CREATE with CONSTRAINT II

- Creating database tables with composite primary keys using PRIMARY KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```
CREATE TABLE EMP_SKILL(
    EmpID    Integer      NOT NULL, 
    SkillID  Integer      NOT NULL, 
    SkillLevel Integer      NULL, 
    CONSTRAINT EmpSkill_PK PRIMARY KEY (EmpID, SkillID)
);
```
SQL for Data Definition: CREATE with CONSTRAINT III

- Creating database tables using PRIMARY KEY and FOREIGN KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword

```sql
CREATE TABLE EMP_SKILL(
    EmpID       Integer      NOT NULL,
    SkillID     Integer      NOT NULL,
    SkillLevel  Integer      NULL,
    CONSTRAINT  EmpSkill_PK PRIMARY KEY
                  (EmpID, SkillID),
    CONSTRAINT  Emp_FK FOREIGN KEY(EmpID)
                  REFERENCES   EMPLOYEE (EmpID),
    CONSTRAINT  Skill_FK FOREIGN KEY(SkillID)
                  REFERENCES   SKILL(SkillID)
);
```

The Entity-Relationship Model 7-9
SQL for Data Definition:
CREATE with CONSTRAINT IV

- Creating database tables using PRIMARY KEY and FOREIGN KEY constraints
  - The SQL CREATE TABLE statement
  - The SQL CONSTRAINT keyword
  - ON UPDATE CASCADE and ON DELETE CASCADE

```sql
CREATE TABLE EMP_SKILL(
    EmpID Integer       NOT NULL,
    SkillID Integer       NOT NULL,
    SkillLevel Integer       NULL,
    CONSTRAINT  EmpSkill_PK   PRIMARY KEY(EmpID, SkillID),
    CONSTRAINT  Emp_FK FOREIGN KEY(EmpID)
        REFERENCES EEMPLOYEE(EmpID)
        ON DELETE CASCADE,
    CONSTRAINT  Skill_FK FOREIGN KEY(SkillID)
        REFERENCES SKILL(SkillID)
        ON UPDATE CASCADE
);
```
Process SQL CREATE TABLE Statements:
Microsoft SQL Server 2012

The SQL script in the tabbed script window.

The objects representing the tables created by the script are shown in the expanded Tables folder—*dbo* stands for *database owner*.

Messages are shown here—either that the commands were successful or appropriate error messages.
Process SQL CREATE TABLE Statements:
Oracle Database 11g Release 2

The SQL script in the tabbed SQL Worksheet window

The objects representing the tables created by the script are shown in the expanded Tables folder—APEX$ tables hold metadata

Messages are shown here—either that the commands were successful or appropriate error messages

The Entity-Relationship Model 7-12
Process SQL CREATE TABLE Statements:
Oracle MySQL 5.5

The SQL script in the tabbed script window

The objects representing the tables created by the script are shown in the expanded _wpc_ schema
Database Diagram in the Microsoft SQL Server Management Studio

The database tables and the links between them are shown in the tabbed Diagram window.

The object representing the database diagram is shown in the expanded Database Diagrams folder—db0 stands for database owner.
Primary Key Constraint: ALTER I

- Adding primary key constraints to an existing table
  - The SQL ALTER statement

```sql
ALTER TABLE EMPLOYEE
  ADD CONSTRAINT Emp_PK PRIMARY KEY(EmpID);
```
Composite Primary Key Constraints: ALTER II

- Adding a composite primary key constraint to an existing table
  - The SQL ALTER statement

```
ALTER TABLE EMP_SKILL
    ADD CONSTRAINT EmpSkill_PK
    PRIMARY KEY(EmpID, SkillID);
```
Foreign Key Constraint: ALTER III

- Adding foreign key constraints to an existing table
  - The SQL ALTER statement

```
ALTER TABLE EMPLOYEE ADD
  CONSTRAINT Emp_FK FOREIGN KEY(DeptID)
  REFERENCES DEPARTMENT(DeptID);
```
Adding Data:

**INSERT**

- To add a row to an existing table, use the `INSERT` statement.
- Non-numeric data must be enclosed in straight (') single quotes.

```
INSERT INTO EMPLOYEE VALUES(91, 'Smither', 12);

INSERT INTO EMPLOYEE (EmpID, SalaryCode)
VALUES (62, 11);
```
SQL for Data Retrieval: Queries

- SELECT is the best known SQL statement.
- SELECT will retrieve information from the database that matches the specified criteria using the SELECT/FROM/WHERE framework.

```
SELECT EmpName
FROM EMPLOYEE
WHERE EmpID = 2010001;
```
SQL for Data Retrieval: The Results of a Query Is a Relation

- A query pulls information from one or more relations and creates (temporarily) a new relation.
- This allows a query to:
  - Create a new relation
  - Feed information to another query (as a “sub-query”)
SQL for Data Retrieval: Displaying All Columns

- To show all of the column values for the rows that match the specified criteria, use an asterisk (*).

```sql
SELECT *  
FROM EMPLOYEE;
```
SQL for Data Retrieval: Showing Each Row Only Once

- The DISTINCT keyword may be added to the SELECT statement to inhibit duplicate rows from displaying.

```sql
SELECT DISTINCT DeptID
FROM EMPLOYEE;
```
SQL for Data Retrieval: Specifying Search Criteria

- The WHERE clause stipulates the matching criteria for the record that is to be displayed.

```
SELECT   EmpName
FROM      EMPLOYEE
WHERE     DeptID = 15;
```
Processing SQL Query Statements: Microsoft SQL Server 2012

The New Query button

The Execute button

The SQL statement in the tabbed query window

The query results

The Entity-Relationship Model 7-24
Processing SQL Query Statements: Oracle Database 11g Release 2

The WPC tabbed SQL Worksheet window

The Run Statement button

The SQL statement in the tabbed SQL Worksheet window

The query results in the tabbed Query Result window
Processing SQL Query Statements: Oracle MySQL 5.5
SQL for Data Retrieval: Match Criteria

The WHERE clause match criteria may include

- Equals “=”
- Not Equals “<>”
- Greater than “>”
- Less than “<”
- Greater than or Equal to “>=”
- Less than or Equal to “<=”
SQL for Data Retrieval: Match Operators

- Multiple matching criteria may be specified using
  - AND
    - Representing an intersection of the data sets
  - OR
    - Representing a union of the data sets
SQL for Data Retrieval: Operator Examples

SELECT EmpName
FROM EMPLOYEE
WHERE DeptID < 7
 OR DeptID > 12;

SELECT EmpName
FROM EMPLOYEE
WHERE DeptID = 9
 AND SalaryCode <= 23;
SQL for Data Retrieval:
A List of Values

- The WHERE clause may include the IN keyword to specify that a particular column value must be included in a list of values.

```
SELECT EmpName
FROM EMPLOYEE
WHERE DeptID IN (4, 8, 9);
```
SQL for Data Retrieval: The Logical NOT Operator

- Any criteria statement may be preceded by a NOT operator, which is to say that all information will be shown except that information matching the specified criteria.

```sql
SELECT EmpName
FROM EMPLOYEE
WHERE DeptID NOT IN (4, 8, 9);
```
SQL for Data Retrieval:
Finding Data in a Range of Values

- SQL provides a BETWEEN keyword that allows a user to specify a minimum and maximum value on one line.

```
SELECT    EmpName
FROM      EMPLOYEE
WHERE     SalaryCode BETWEEN 10 AND 45;
```
SQL for Data Retrieval: Allowing for Wildcard Searches

- The SQL LIKE keyword allows searches on partial data values.
- LIKE can be paired with wildcards to find rows matching a string value.
  - Multiple character wildcard character is a percent sign (%).
  - Single character wildcard character is an underscore (_).
SQL for Data Retrieval:
Wildcard Search Examples

SELECT EmpID
FROM EMPLOYEE
WHERE EmpName LIKE 'Kr%';

SELECT EmpID
FROM EMPLOYEE
WHERE Phone LIKE '616-____-____';
SQL for Data Retrieval: Sorting the Results

- Query results may be sorted using the ORDER BY clause.

```sql
SELECT * 
FROM EMPLOYEE 
ORDER BY EmpName;
```
SQL for Data Retrieval:
Built-in SQL Functions

- SQL provides several built-in functions:
  - **COUNT**
    - Counts the number of rows that match the specified criteria
  - **MIN**
    - Finds the minimum value for a specific column for those rows matching the criteria
  - **MAX**
    - Finds the maximum value for a specific column for those rows matching the criteria
SQL for Data Retrieval: Built-in SQL Functions (Cont’d)

- **SUM**
  - Calculates the sum for a specific column for those rows matching the criteria

- **AVG**
  - Calculates the numerical average of a specific column for those rows matching the criteria
SQL for Data Retrieval:
Built-in Function Examples

```
SELECT COUNT(DeptID)
FROM EMPLOYEE;

SELECT MIN(Hours) AS MinimumHours,
       MAX(Hours) AS MaximumHours,
       AVG(Hours) AS AverageHours
FROM PROJECT
WHERE ProjID > 7;
```
SQL for Data Retrieval:
Providing Subtotals: GROUP BY

- Subtotals may be calculated by using the GROUP BY clause.
- The HAVING clause may be used to restrict which data is displayed.

```sql
SELECT DeptID,
       COUNT(*) AS NumOfEmployees
FROM EMPLOYEE
GROUP BY DeptID
HAVING COUNT(*) > 3;
```
SQL for Data Retrieval: Retrieving Information from Multiple Tables

- Subqueries
  - As stated earlier, the result of a query is a relation. As a result, a query may feed another query. This is called a subquery.

- Joins
  - Another way of combining data is by using a join.
    - Join [also called an Inner Join]
    - Left Outer Join
    - Right Outer Join
SELECT EmpName
FROM EMPLOYEE
WHERE DeptID in
    (SELECT DeptID
     FROM DEPARTMENT
     WHERE DeptName LIKE 'Account%');
SQL for Data Retrieval:
Join Example

```
SELECT  EmpName
FROM    EMPLOYEE AS E, DEPARTMENT AS D
WHERE   E.DeptID = D.DeptID
        AND  D.DeptName LIKE 'Account%';
```
SQL for Data Retrieval: JOIN…ON Example

- The JOIN…ON syntax can be used in joins.
- It has the advantage of moving the JOIN syntax into the FROM clause.

```
SELECT EmpName
FROM EMPLOYEE AS E JOIN DEPARTMENT AS D
  ON E.DeptID = D.DeptID
WHERE D.DeptName LIKE 'Account%';
```
SQL for Data Retrieval: LEFT OUTER JOIN Example

- The OUTER JOIN syntax can be used to obtain data that exists in one table without matching data in the other table.

```sql
SELECT EmpName
FROM EMPLOYEE AS E
  LEFT JOIN DEPARTMENT AS D
  ON E.DeptID = D.DeptID
WHERE D.DeptName LIKE 'Account%';
```
SQL for Data Retrieval: 
RIGHT OUTER JOIN Example

- The unmatched data displayed can be from either table, depending on whether RIGHT JOIN or LEFT JOIN is used.

```sql
SELECT  EmpName
FROM    EMPLOYEE AS E
RIGHT JOIN DEPARTMENT AS D
    ON  E.DeptID = D.DeptID
WHERE   D.DeptName LIKE 'Account%';
```
Modifying Data using SQL

- **Insert**
  - Will add a new row in a table (already discussed above)

- **Update**
  - Will update the data in a table that matches the specified criteria

- **Delete**
  - Will delete the data in a table that matches the specified criteria
Modifying Data using SQL:
Changing Data Values: UPDATE

- To change the data values in an existing row (or set of rows) use the Update statement.

```
UPDATE EMPLOYEE
SET Phone '791-555-1234'
WHERE EmpID = 29;

UPDATE EMPLOYEE
SET DeptID = 44
WHERE EmpName LIKE 'Kr%';
```
Modifying Data using SQL: MERGE

- SQL:2003 introduced the MERGE statement.
  - Combines INSERT and UPDATE into one statement
  - Uses the equivalent of IF-THEN-ELSE logic to decide whether to use INSERT or UPDATE
  - An advanced feature—learn to use INSERT and UPDATE separately first, then consult DBMS documentation
Modifying Data using SQL: Deleting Data: DELETE

- To delete a row or set of rows from a table use the DELETE statement.

```
DELETE FROM EMPLOYEE
WHERE EmpID = 29;
```

```
DELETE FROM EMPLOYEE
WHERE EmpName LIKE 'Kr%';
```
Modifying Data using SQL:
Deleting Database Objects: DROP

- To remove unwanted database objects from the database, use the SQL DROP statement.
- Warning… The DROP statement will permanently remove the object and all data.

```
DROP TABLE EMPLOYEE;
```
To change the constraints on existing tables, you may need to remove the existing constraints before adding new constraints.

```
ALTER TABLE EMPLOYEE DROP CONSTRAINT EmpFK;
```
Modifying Data Using SQL: The CHECK Constraint

- The CHECK constraint can be used to create sets of values to restrict the values that can be used in a column.

```
ALTER TABLE PROJECT
    ADD CONSTRAINT PROJECT_Check_Dates
    CHECK (StartDate < EndDate);
```
SQL Views

- A SQL View is a virtual table created by a DBMS-stored SELECT statement that can combine access to data in multiple tables and even in other views.
- SQL views are discussed online in Appendix E.
Chapter 7: outline

7.1 SQL
7.2 The Entity-Relationship Model
Three Stages of Database Development

- The three stages of database development are:
  - Requirements Analysis Stage
  - Component Design Stage
  - Implementation Stage
The Requirements Analysis Stage

- Sources of requirements
  - User Interviews
  - Forms
  - Reports
  - Queries
  - Use Cases
  - Business Rules
Requirements Become the E-R Data Model

- After the requirements have been gathered, they are transformed into an Entity Relationship (E-R) Data Model.
- The most important elements of E-R Models are:
  - Entities
  - Attributes
  - Identifiers
  - Relationships
Entity Class versus Entity Instance

- An **entity class** is a description of the structure and format of the occurrences of the entity.
- An **entity instance** is a specific occurrence of an entity within an entity class.
Entity Class and Entity Instance

**Entity Class**

<table>
<thead>
<tr>
<th>ITEM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ItemNumber</td>
</tr>
<tr>
<td>Description</td>
</tr>
<tr>
<td>Cost</td>
</tr>
<tr>
<td>ListPrice</td>
</tr>
<tr>
<td>QuantityOnHand</td>
</tr>
</tbody>
</table>

**Two Entity Instances**

1. 1100
   - 100 amp panel
   - $127.50
   - $170.00
   - 14

2. 2000
   - Door handle set
   - $52.50
   - $39.38
   - 0
Attributes

- Entities have **attributes** that describe the entity’s characteristics:
  - ProjectName
  - StartDate
  - ProjectType
  - ProjectDescription

- Attributes have a data type and properties.
Identifiers

- Entity instances have identifiers.
- An identifier will identify a particular instance in the entity class:
  - SocialSecurityNumber
  - StudentID
  - EmployeeID
Identifier Types

- **Uniqueness**
  - Identifiers may be **unique** or **nonunique**.
  - If the identifier is unique, the data value for the identifier must be unique for all instances.

- **Composite**
  - A **composite identifier** consists of two or more attributes.
    - E.g., *OrderNumber & LineItemNumber* are both required.
Levels of Entity Attribute Display

(a) Entity with All Attributes

(b) Entity with Identifier Attribute Only

(c) Entity with No Attributes

The Entity-Relationship Model 7-63
Relationships

- Entities can be associated with one another in relationships.
- Relationship degree defines the number of entity classes participating in the relationship:
  - Degree 2 is a binary relationship.
  - Degree 3 is a ternary relationship.
Degree 2 Relationship: Binary

(a) Binary Relationship
Degree 3 Relationship: Ternary

(b) Ternary Relationship
One-to-One Binary Relationship

- **1:1 (one-to-one)**
  - A single entity instance in one entity class is related to a single entity instance in another entity class.
    - An employee may have no more than one locker; and
    - A locker may only be accessible by one employee
One-to-Many Binary Relationship

- **1:N (one-to-many)**
  - A single entity instance in one entity class is related to many entity instances in another entity class.
    - A quotation is associated with only one item; and
    - An item may have several quotations
Many-to-Many Binary Relationship

- **N:M (many-to-many)**
  - Many entity instances in one entity class is related to many entity instances in another entity class:
    - a supplier may supply several items; and
    - a particular item may be supplied by several suppliers.
Maximum Cardinality

- Relationships are named and classified by their **cardinality**, which is a word that means *count*.
- Each of the three types of binary relationships shown above have different *maximum cardinalities*.
- **Maximum cardinality** is the maximum number of entity instances that may participate in a relationship instance—one, many, or some other fixed number.
Minimum Cardinality

- **Minimum cardinality** is the minimum number of entity instances that *must* participate in a relationship instance.
- These values typically assume a value of zero (optional) or one (mandatory).
Cardinality Example

- Maximum cardinality is many for both ITEM and SUPPLIER.
- Minimum cardinality is zero (optional) for ITEM and one (mandatory) SUPPLIER.
  - A SUPPLIER does not have to supply an ITEM.
  - An ITEM must have a SUPPLIER.
Entity-Relationship Diagrams

- The diagrams in previous slides are called **entity-relationship diagrams**.
  - Entity classes are shown by rectangles.
  - Relationships are shown by diamonds.
  - The maximum cardinality of the relationship is shown inside the diamond.
  - The minimum cardinality is shown by the oval or hash mark next to the entity.
  - The name of the entity is shown inside the rectangle.
  - The name of the relationship is shown near the diamond.
HAS-A Relationships

- The relationships in the previous slides are called HAS-A relationships.
- The term is used because each entity instance has a relationship to a second entity instance:
  - An employee has a badge.
  - A badge has an employee.
Types of Entity-Relationship Diagrams

- **Information Engineering (IE) [James Martin 1990]**—Uses “crow’s feet” to show the many sides of a relationship, and it is sometimes called the crow’s foot model.

- **Integrated Definition 1, Extended 3 (IDEFIX)**

- **Unified Modeling Language (UML)** is a set of structures and techniques for modeling and designing object-oriented programs (OOP) and applications
Crow’s Foot Example: One-to-Many Relationship

(a) Original E-R Model Version

(b) Crow’s Foot Version
## Crow’s Foot Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Mandatory-One" /></td>
<td>Mandatory—One</td>
</tr>
<tr>
<td><img src="image" alt="Mandatory-Many" /></td>
<td>Mandatory—Many</td>
</tr>
<tr>
<td><img src="image" alt="Optional-One" /></td>
<td>Optional—One</td>
</tr>
<tr>
<td><img src="image" alt="Optional-Many" /></td>
<td>Optional—Many</td>
</tr>
</tbody>
</table>
Crow’s Foot Example: Many-to-Many Relationship

(a) Original E-R Model Version

(b) Crow’s Foot Version
A weak entity is an entity that cannot exist in the database without the existence of another entity.

Any entity that is not a weak entity is called a strong entity.
ID-Dependent Weak Entities

- An ID-Dependent weak entity is a weak entity that cannot exist without its parent entity.
- An ID-dependent weak entity has a composite identifier.
  - The first part of the identifier is the identifier for the strong entity.
  - The second part of the identifier is the identifier for the weak entity itself.
ID-Dependent Weak Entity Examples

- **BUILDING**
  - BuildingName
  - StreetAddress
  - City
  - State
  - ZIP

- **PRODUCT**
  - ProductName
  - OperatingSystem
  - DevTeamEmail
  - DevTeamPhone

- **TEXTBOOK**
  - Title
  - Author
  - Publisher

- **APARTMENT**
  - BuildingName
  - ApartmentNumber
  - NumberOfBedrooms
  - NumberOfBaths
  - MonthlyRent

- **VERSION**
  - ProductName
  - VersionNumber
  - ReleaseDate
  - MemoryRequired
  - DiskSpaceRequired

- **EDITION**
  - Title
  - EditionNumber
  - ISBN
  - CopyrightDate
  - NumberOfPages

(a) APARTMENT is ID-Dependent on BUILDING
(b) VERSION is ID-Dependent on PRODUCT
(c) EDITION is ID-Dependent on TEXTBOOK
Weak Entity Relationships

- The relationship between a strong and weak entity is termed an **identifying relationship** if the weak entity is ID-dependent.
  - Represented by a *solid line*

- The relationship between a strong and weak entity is termed a **nonidentifying relationship** if the weak entity is non-ID-dependent.
  - Represented by a *dashed line*
  - Also used between strong entities
Weak Entity Identifier: Non-ID-dependent

- All ID-dependent entities are weak entities, but there are other entities that are weak but not ID-dependent.
- A non-ID-dependent weak entity may have a single or composite identifier, but the identifier of the parent entity will be a *foreign key*.
Non-ID-Dependent Weak Entity Examples

(a) ID-Dependent Entity

(b) Non-ID-Dependent Weak Entity
Strong and Weak Entity Examples

(a) ORDER is a Strong Entity

(b) ASSIGNMENT is an ID-Dependent Entity

(c) PRESCRIPTION is a Non-ID-Dependent Weak Entity
Subtype Entities

- A **subtype** entity is a special case of another entity called **supertype**.
- An attribute of the supertype may be included that indicates which of the subtypes is appropriate for a given instance; this attribute is called a **discriminator**.
- Subtypes can be **exclusive** or **inclusive**.
  - If **exclusive**, the supertype relates to at most one subtype.
  - If **inclusive**, the supertype can relate to one or more subtypes.
The relationships that connect supertypes and subtypes are called **IS-A relationships** because a subtype is the same entity as the supertype.

- The identifier of a supertype and all of its subtypes is the same attribute.
Subtype Entity Examples

(a) Exclusive Subtypes with Discriminator

(b) Inclusive Subtypes
Recursive Relationships

- It is possible for an entity to have a relationship to itself—this is called a recursive relationship.
Developing an E-R Diagram

- Heather Sweeney Designs will be used as an ongoing example throughout Chapters 4, 5, 6, and 7.
  - Heather Sweeney is an interior designer who specializes in home kitchen design.
  - She offers a variety of free seminars at home shows, kitchen and appliance stores, and other public locations.
  - She earns revenue by selling books and videos that instruct people on kitchen design.
  - She also offers custom-design consulting services.
# Heather Sweeney Designs: The Seminar Customer List

<table>
<thead>
<tr>
<th>Name</th>
<th>Phone</th>
<th>Email Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nancy Jacobs</td>
<td>817-871-8123</td>
<td><a href="mailto:NJ@somewhere.com">NJ@somewhere.com</a></td>
</tr>
<tr>
<td>Chantel Jacobs</td>
<td>817-871-8234</td>
<td><a href="mailto:CJ@somewhere.com">CJ@somewhere.com</a></td>
</tr>
<tr>
<td>Ralph Able</td>
<td>210-281-7687</td>
<td><a href="mailto:RA@somewhere.com">RA@somewhere.com</a></td>
</tr>
<tr>
<td>Etc.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

27 names in all
Heather Sweeney Designs: Initial E-R Diagram I

SEMINAR
- ??? is SEMINAR identifier
- SeminarDate
- SeminarTime
- Location
- SeminarTitle

Is this correct?

CUSTOMER
- ??? is CUSTOMER identifier
- LastName
- FirstName
- Phone
- EmailAddress

Is this correct?
Heather Sweeney Designs:
Initial E-R Diagram II

SEMINAR
- SeminarDate
- SeminarTime
- Location
- SeminarTitle

CUSTOMER
- LastName
- FirstName
- Phone
- EmailAddress

The Entity-Relationship Model 7-93
Heather Sweeney Designs:
Initial E-R Diagram III

SEMINAR
- SeminarID
- SeminarDate
- SeminarTime
- Location
- SeminarTitle

CUSTOMER
- EmailAddress
- LastName
- FirstName
- Phone
Heather Sweeney Designs:  
The Customer Form Letter

Heather Sweeney Designs  
122450 Rockaway Road  
Dallas, Texas 75227  
972-233-6165

Ms. Nancy Jacobs  
1400 West Palm Drive  
Fort Worth, Texas 76110

Dear Ms. Jacobs:

Thank you for attending my seminar “Kitchen on a Budget” at the San Antonio Convention Center. I hope that you found the seminar topic interesting and helpful for your design projects.

As a seminar attendee, you are entitled to a 15 percent discount on all of my video and book products. I am enclosing a product catalog and I would also like to invite you to visit our Web site at www.Sweeney.com.

Also, as I mentioned at the seminar, I do provide customized design services to help you create that just-perfect kitchen. In fact, I have a number of clients in the Fort Worth area. Just give me a call at my personal phone number of 855-122-4873 if you’d like to schedule an appointment.

Thanks again and I look forward to hearing from you!

Best regards,

Heather Sweeney
Heather Sweeney Designs:
Data Model with CONTACT
Heather Sweeney Designs: Data Model with CONTACT as Weak Entity
Heather Sweeney Designs: Data Model with Modified CUSTOMER

The Entity-Relationship Model 7-98
Heather Sweeney Designs: Sales Invoice

<table>
<thead>
<tr>
<th>Customer</th>
<th>Date</th>
<th>Invoice No.</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name: Ralph Able</td>
<td>10/18/12</td>
<td>35000</td>
<td>$29.54</td>
</tr>
<tr>
<td>Address: 133 Elm Street</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>City: San Antonio</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZIP: 76214</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phone: 210-281-1987</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Qty</th>
<th>Description</th>
<th>Unit Price</th>
<th>TOTAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Kitchen Remodeling Basics - Video</td>
<td>$14.95</td>
<td>$14.95</td>
</tr>
<tr>
<td>1</td>
<td>Kitchen Remodeling Basics - Video Companion</td>
<td>$7.00</td>
<td>$7.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Payment</th>
<th>Tax Rate(s)</th>
<th>Subtotal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credit</td>
<td></td>
<td>$29.54</td>
</tr>
<tr>
<td></td>
<td>5.70%</td>
<td>$1.31</td>
</tr>
</tbody>
</table>

TOTAL $30.20
Heather Sweeney Designs: Data Model with INVOICE

The Entity-Relationship Model 7-100
Heather Sweeney Designs: Data Model with LINE_ITEM
Heather Sweeney Designs:
Final Data Model

The Entity-Relationship Model 7-102
Heather Sweeney Designs: Business Rules and Model Validation

- **Business rules** may constrain the model and need to be recorded.
  - Heather Sweeney Designs has a business rule that no more than one form letter or email per day is to be sent to a customer.

- After the data model has been completed, it needs to be validated.
  - **Prototyping** is commonly used to validate forms and reports.

The Entity-Relationship Model 7-103
Chapter 7: summary

- Structured Query Language
  - data definition
  - keys
  - data retrieval
  - modifying data

- The Entity-Relationship Model
  - analysis of requirements
  - component design
  - implementation