



**DAVID M. KROENKE and DAVID J. AUER**  
**DATABASE CONCEPTS, 4<sup>th</sup> Edition**



# Database Design

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems



## Chapter Objectives



- Learn how to transform E-R data models into relational designs
- Practice the normalization process
- Understand the need for de-normalization
- Learn how to represent weak entities with the relational model
- Know how to represent 1:1, 1:N, and N:M binary relationships
- Know how to represent 1:1, 1:N, and N:M recursive relationships
- Learn SQL statements for creating joins over binary and recursive relationships
- Understand the nature and background of normalization

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-2



## Transforming a Data Model into a Relational Design

1. Create a table for each entity:
  - Specify primary key (consider surrogate keys as appropriate)
  - Specify properties for each column:
    - Data type
    - Null status
    - Default value (if any)
    - Specify data constraints (if any)
  - Verify normalization
2. Create relationships by placing foreign keys:
  - Strong entity relationships (1:1, 1:N, N:M)
  - ID-dependent and non-ID-dependent weak entity relationships
  - Subtypes
  - Recursive (1:1, 1:N, N:M)



## Representing Entities with the Relational Model

- Create a relation for each entity
  - A relation has a descriptive name and a set of attributes that describe the entity
- Specify a primary key
- Specify column properties
  - Data type
  - Null status
  - Default values (if any)
  - Data constraints (if any)
- The relation is then analyzed using the normalization rules
- As normalization issues arise, the initial relation design may need to change



## Representing an Entity as a Table

ITEM

ItemNumber
Description
Cost
ListPrice
QuantityOnHand

(a) The ITEM Entity


ITEM

 ItemNumber
Description
Cost
ListPrice
QuantityOnHand

(b) The ITEM Table

ITEM (ItemNumber, Description, Cost, ListPrice, QuantityOnHand)

ITEM

 ItemNumber: int IDENTITY(10000,1)
Description: varchar(100) NOT NULL
Cost: numeric(9,2) NOT NULL
ListPrice: numeric(9,2) NULL
QuantityOnHand: int NOT NULL

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-5



## Normalization Review: Modification Problems

- Tables that are not normalized will experience issues known as modification problems
  - Insertion problems
    - ✓ Difficulties inserting data into a relation
  - Modification problems
    - ✓ Difficulties modifying data into a relation
  - Deletion problems
    - ✓ Difficulties deleting data from a relation

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-6



## Normalization Review: Solving Modification Problems

- Most modification problems are solved by breaking an existing table into two or more tables through a process known as normalization



## Normalization Review: Definition Review

- **Functional dependency**
  - The relationship (within the relation) that describes how the value of one attribute may be used to find the value of another attribute
- **Determinant**
  - The attribute that can be used to find the value of another attribute in the relation
  - The right-hand side of a functional dependency



## Normalization Review: Definition Review II

- Candidate key
  - The value of a candidate key can be used to find the value of every other attribute in the table
  - A simple candidate key consists of only one attribute
  - A composite candidate key consists of more than one attribute



## Normalization Review: Normal Forms

- There are many defined normal forms:
  - First Normal Form (1NF)
  - Second Normal Form (2NF)
  - Third Normal Form (3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth Normal Form (4NF)
  - Fifth Normal Form (5NF)
  - Domain/Key Normal Form (DK/NF)

- Rows contain data about an entity.
- Columns contain data about attributes of the entity.
- Cells of the table hold a single value.
- All entries in a column are of the same kind.
- Each column has a unique name.
- The order of the columns is unimportant.
- The order of the rows is unimportant.
- No two rows may be identical



## Normalization Review: Normalization

- For our purposes, a relation is considered normalized when:

Every determinant is a candidate key

[Technically, this is Boyce-Codd Normal Form (BCNF)]



## The CUSTOMER Table

CUSTOMER
CustomerNumber
CustomerName
StreetAddress
City
State
ZIP
ContactName
Phone

(a) The CUSTOMER Entity

CUSTOMER
CustomerNumber
CustomerName
StreetAddress
City
State
ZIP
ContactName
Phone

(b) The CUSTOMER Table

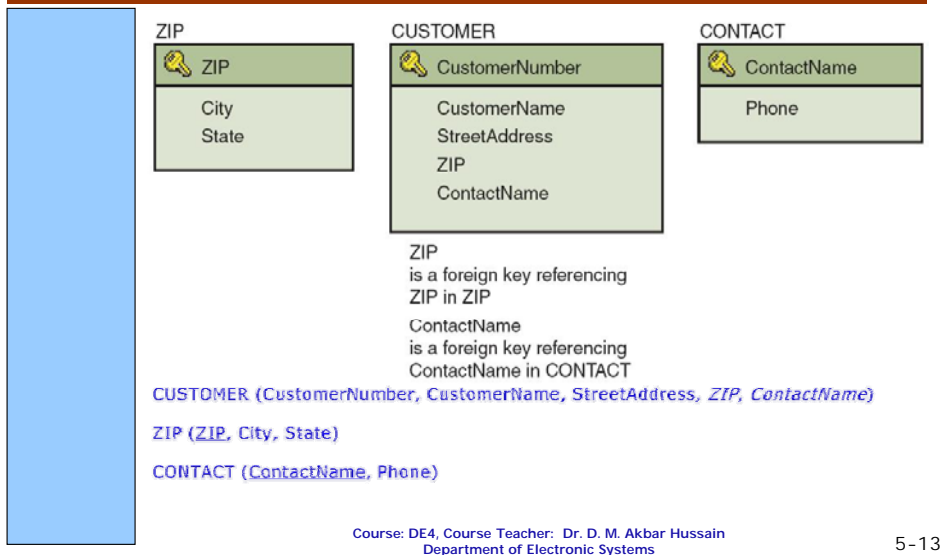
**CUSTOMER** (CustomerNumber, CustomerName, StreetAddress, City, State, ZIP, ContactName, Phone)

*ZIP* → (*City*, *State*)

*ContactName* → *Phone*



## The CUSTOMER Entity: The Normalized Set of Tables



## Denormalization

- Normalizing relations (or breaking them apart into many component relations) may significantly increase the complexity of the data structure
- The question is one of balance
  - Trading complexity for modification problems
- There are situations where denormalized relations are preferred




## The CUSTOMER Entity: The De-normalized Set of Tables

### CUSTOMER

 CustomerNumber
CustomerName StreetAddress City State ZIP ContactName

### CONTACT

 ContactName
Phone

ContactName  
is a foreign key referencing  
ContactName in CONTACT

**CUSTOMER** (CustomerNumber, CustomerName, StreetAddress, City, State, ZIP, ContactName)

**CONTACT** (ContactName, Phone)




## SALES\_COMMISSION Entity and Table

### SALES\_COMMISSION

CheckNumber SalespersonNumber SalespersonLastName SalespersonFirstName Phone CheckDate CommissionPeriod TotalCommissionSales CommissionAmount BudgetCategory
---

### SALES\_COMMISSION

 CheckNumber
SalespersonNumber SalespersonLastName SalespersonFirstName Phone CheckDate CommissionPeriod TotalCommissionSales CommissionAmount BudgetCategory

**SALES\_COMMISSION** (SalespersonNumber, SalespersonLastName, SalespersonFirstName, Phone, CheckNumber, CheckDate, CommissionPeriod, TotalCommissionSales, CommissionAmount, BudgetCategory)

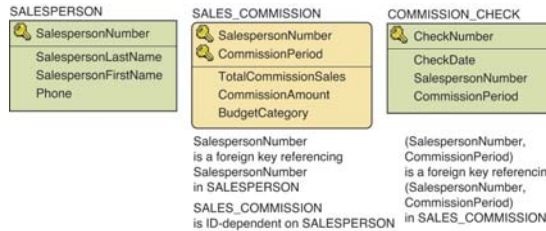
**SalespersonNumber** → (SalespersonLastName, SalespersonFirstName, Phone, BudgetCategory)

**SalespersonNumber, CommissionPeriod** → (TotalCommissionSales, CommissionAmount)





## Normalized



**SALES\_COMMISSION** (*SalespersonNumber*, *CommissionPeriod*, TotalCommissionSales, CommissionAmount, BudgetCategory)

**SALESPERSON**(*SalespersonNumber*, SalespersonLastName, SalespersonFirstName, Phone)

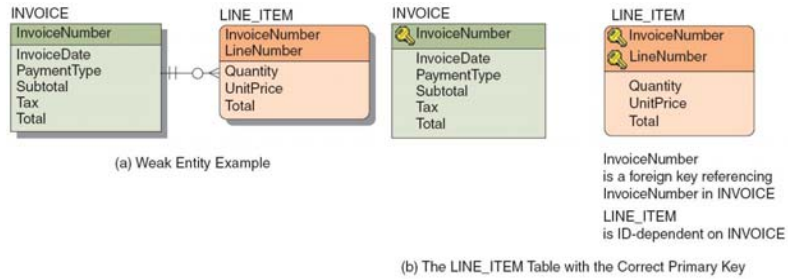
**COMMISSION\_CHECK** (*CheckNumber*, CheckDate, *SalespersonNumber*, *CommissionPeriod*)



## Representing Weak Entities

- If not ID-dependent, use the same techniques as for strong entities.
- If ID-dependent, then must add primary key of the parent entity.

## Representing Weak Entities Example

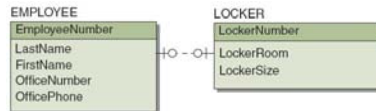


## Representing Relationships 1:1 Relationships

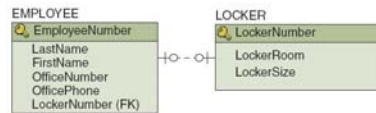
- The maximum cardinality determines how a relationship is represented
- 1:1 relationship
  - The key from one relation is placed in the other as a foreign key.
  - It does not matter which table receives the foreign key.



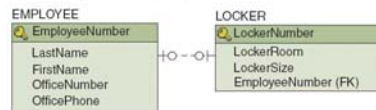
## Representing Relationships 1:1 Relationship Example



(a) Example 1:1 Strong Entity Relationship



(b) Placing the Primary Key of LOCKER into EMPLOYEE



(c) Placing the Primary Key of EMPLOYEE into LOCKER

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-21



## Representing Relationships SQL for 1:1 Relationships

```
SELECT      *
FROM        LOCKER, EMPLOYEE
WHERE       LOCKER.LockerNumber =
            EMPLOYEE.LockerNumber;
```

```
SELECT      *
FROM        LOCKER, EMPLOYEE
WHERE       LOCKER.EmployeeNumber =
            EMPLOYEE.EmployeeNumber;
```

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-22

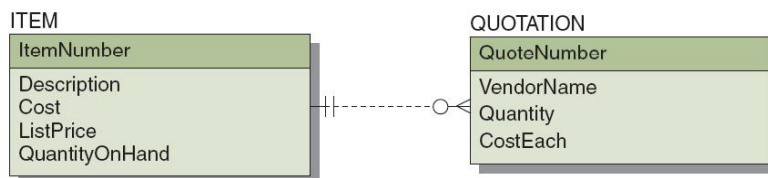


## Representing Relationships 1:N Relationships

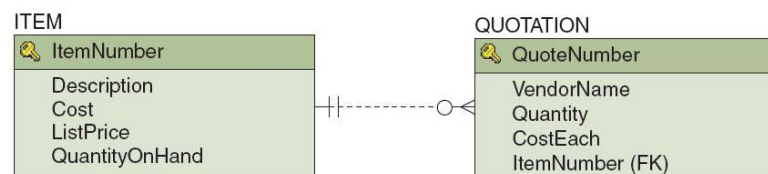
- Like a 1:1 relationship, a 1:N relationship is saved by placing the key from one table into another as a foreign key
- However, in a 1:N the foreign key always goes into the many-side of the relationship
  - The 1 side is called the parent
  - The N side is called the child



## Representing Relationships 1:N Relationship Example



(a) 1:N Strong Entity Relationship Example



(b) Placing the Primary Key of ITEM into QUOTATION

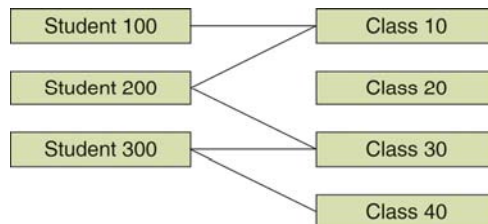
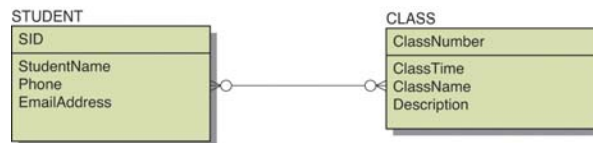


## Representing Relationships SQL for 1:N Relationships

```
SELECT      *
FROM        ITEM, QUOTATION
WHERE       ITEM.ItemNumber =
           QUOTATION.ItemNumber;
```



## Representing Relationships N:M Relationship Example



## Incorrect N:M Relationship

SID	Other STUDENT Data
100	...
200	...
300	...

STUDENT

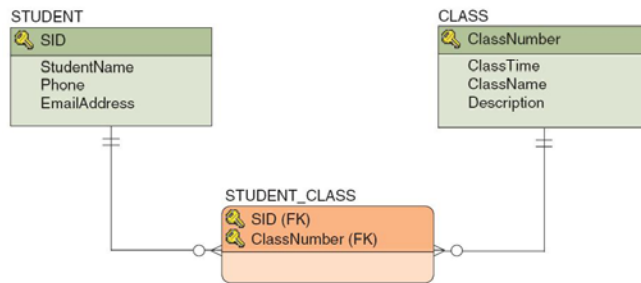
ClassNumber	ClassTime	Other CLASS Data	SID
10	10:00 MWF	...	100
10	10:00 MWF	...	200
30	3:00 TH	...	200
30	3:00 TH	...	300
40	8:00 MWF	...	300

CLASS

## Representing Relationships N:M Relationships

- To create an N:M relationship, a new table is created. This table is called an **intersection table**.
- An **intersection table** has a composite key consisting of the keys from each of the tables that it connects

## Representing Relationships N:M Relationship Example



(a) The STUDENT\_CLASS Intersection Table

100	Jones, Mary	100	10	10	Accounting
200	Parker, Fred	200	10	20	Finance
300	Wu, Jason	200	30	30	Marketing
		300	30	40	Database
		300	40		

(b) Sample Data for the STUDENT-to-CLASS Relationship

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

## Representing Relationships SQL for N:M Relationships

```

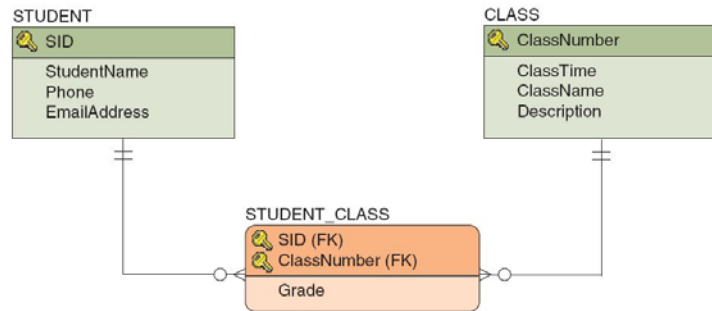
SELECT      *
FROM        STUDENT, CLASS, STUDENT_CLASS
WHERE      STUDENT.SID = STUDENT_CLASS.SID
AND        STUDENT_CLASS.ClassNumber =
           CLASS.ClassNumber;
    
```

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems



## Representing Relationships Association Relationships

- When an intersection table has columns beyond those in the primary key, the relationship is called an **association relationship**



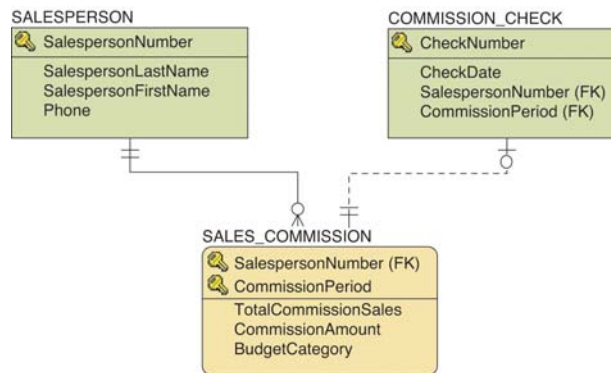
Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-31



## Representing Relationships Mixed Entity Pattern Relationship

- Note there is **non-identifying (1:1)** relation between COMMISSION\_CHECK & SALES\_COMMISSION, there is **1:N** relationship between SALESPERSON & SALES\_COMMISSION.



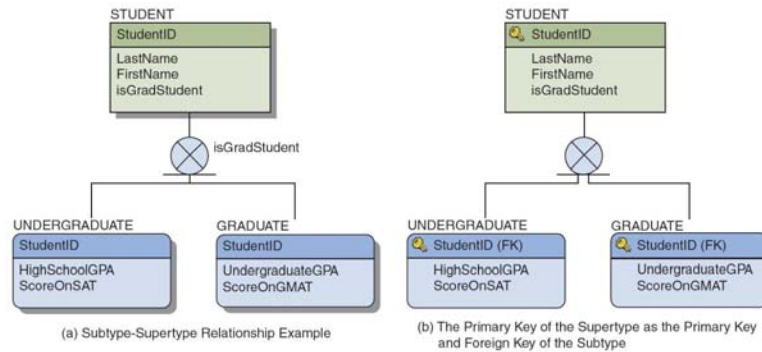
Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-32



## Representing Relationships Supertype/Subtype Relationships

- The identifier of the supertype becomes the primary key and the foreign key of each subtype



Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-33

## Representing Relationships Recursive Relationships

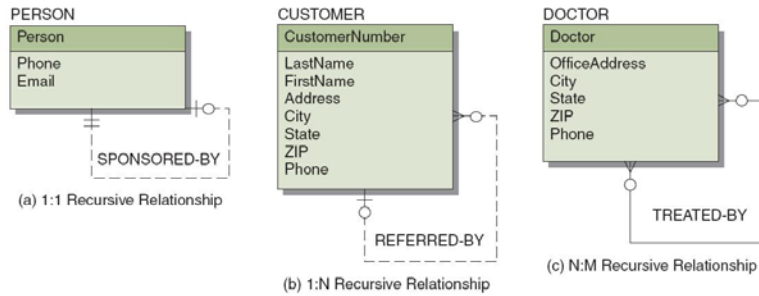
- A recursive relationship is a relationship that a relation has with itself.
- Recursive relationships adhere to the same rules as the binary relationships.
  - 1:1 and 1:N relationships are saved using foreign keys
  - N:M relationships are saved by creating an intersecting relation

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-34

## Representing Relationships

### Recursive Relationships Examples



Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-35

## Representing Relationships

### 1:1 Recursive Relationship Examples

Person

- ↙ Jones
- ↙ Smith
- ↙ Parks
- ↙ Myrtle
- ↙ Pines

PERSON1 Relation

Person	PersonSponsored
Jones	Smith
Smith	Parks
Parks	null
Myrtle	Pines
Pines	null

Referential integrity constraint:  
PersonSponsored in PERSON1  
must exist in Person in PERSON1

```
SELECT *
FROM PERSON1 AS A, PERSON1 AS B
WHERE A.Person = B.PersonSponsored;
```

PERSON2 Relation

Person	PersonSponsoredBy
Jones	null
Smith	Jones
Parks	Smith
Myrtle	null
Pines	Myrtle

Referential integrity constraint:  
PersonSponsoredBy PERSON2  
must exist in Person in PERSON2

```
SELECT *
FROM PERSON2 AS C, PERSON2 AS D
WHERE C.Person = D.PersonSponsoredBy;
```

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-36



## Representing Relationships

### 1:N Recursive Relationship Example



CUSTOMER Relation

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

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

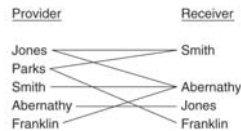
Referential integrity constraint:  
 ReferredBy in CUSTOMER must exist in  
 CustomerNumber in CUSTOMER

```
SELECT *
FROM CUSTOMER AS A, CUSTOMER AS B
WHERE A.CustomerNumber = B.ReferredBy;
```



## Representing Relationships

### N:M Recursive Relationship Example



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

```
SELECT *
FROM DOCTOR AS A, DOCTOR AS B, TREATMENT-INTERSECTION
WHERE A.Name = TREATMENT-INTERSECTION.Physician
AND TREATMENT-INTERSECTION.Patient = B.Name;
```

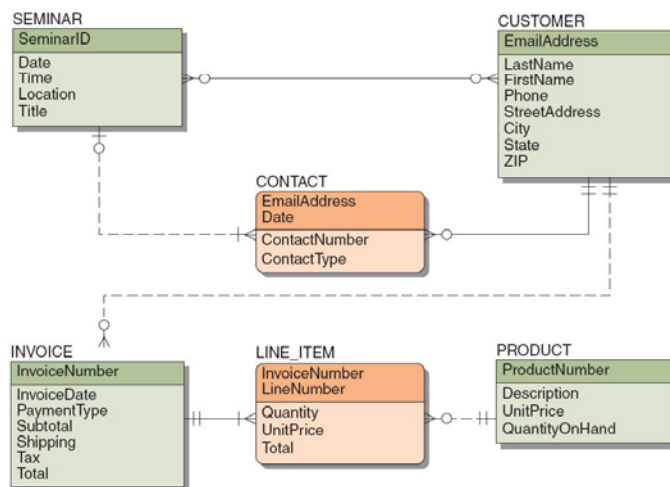


## Heather Sweeney Designs: Developing a Database Design

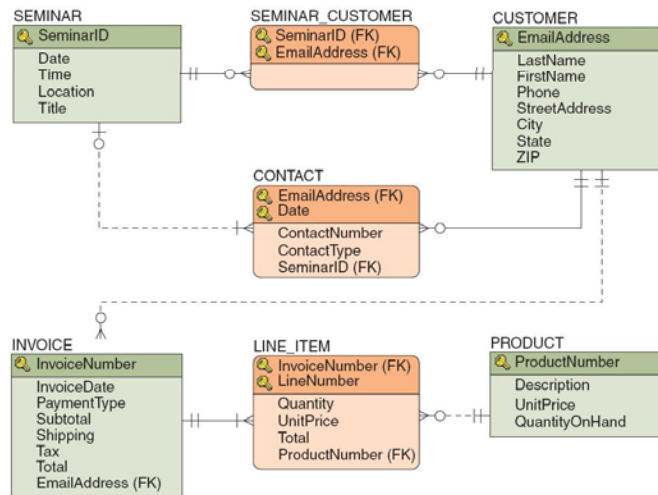
- Heather Sweeney Designs will be used as an ongoing example throughout Chapters 4, 5, 6 and 7:
  - Heather Sweeney is an interior designer who specializes in home kitchen design
  - She offers a variety of free seminars at home shows, kitchen and appliance stores, and other public locations
  - She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



## Heather Sweeney Designs: Final Data Model



## Heather Sweeney Designs: Database Design



Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-41

## Heather Sweeney Designs: Database Design Schema

**SEMINAR** (SeminarID, Date, Time, Location, Title)

**CUSTOMER** (EmailAddress, Name, Phone, Street, City, State, Zip)

**SEMINAR\_CUSTOMER** (SeminarID, EmailAddress)

**CONTACT** (EmailAddress, Date, ContactNumber, ContactType, SeminarID)

**PRODUCT** (ProductNumber, Description, UnitPrice, QuantityOnHand)

**INVOICE** (InvoiceNumber, Date, PaymentType, SubTotal, Tax, Total, EmailAddress)

**LINE\_ITEM** (InvoiceNumber, LineNumber, Quantity, UnitPrice, Total, ProductNumber)

[Referential integrity constraints are in a separate slide]

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-42



## Heather Sweeney Designs: Referential Integrity Constraints

RELATIONSHIP		REFERENTIAL INTEGRITY CONSTRAINT	CASCADING BEHAVIOR	
PARENT	CHILD		ON UPDATE	ON DELETE
SEMINAR	SEMINAR_CUSTOMER	SeminarID in SEMINAR_CUSTOMER must exist in SeminarID in SEMINAR	No	Yes
CUSTOMER	SEMINAR_CUSTOMER	EmailAddress in SEMINAR_CUSTOMER must exist in EmailAddress in CUSTOMER	Yes	Yes
SEMINAR	CONTACT	SeminarID in CONTACT must exist in SeminarID in SEMINAR	No	No
CUSTOMER	CONTACT	EmailAddress in CONTACT must exist in EmailAddress in CUSTOMER	Yes	Yes
CUSTOMER	INVOICE	EmailAddress in INVOICE must exist in EmailAddress in CUSTOMER	Yes	No
INVOICE	LINE_ITEM	InvoiceNumber in LINE_ITEM must exist in InvoiceNumber in INVOICE	No	Yes
PRODUCT	LINE_ITEM	ProductNumber in LINE_ITEM must exist in ProductNumber in PRODUCT	Yes	No

Course: DE4, Course Teacher: Dr. D. M. Akbar Hussain  
Department of Electronic Systems

5-43