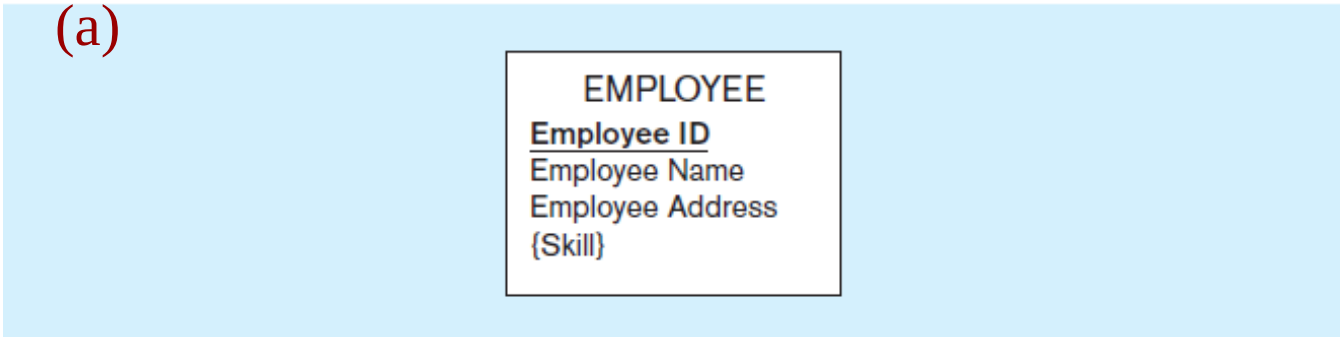
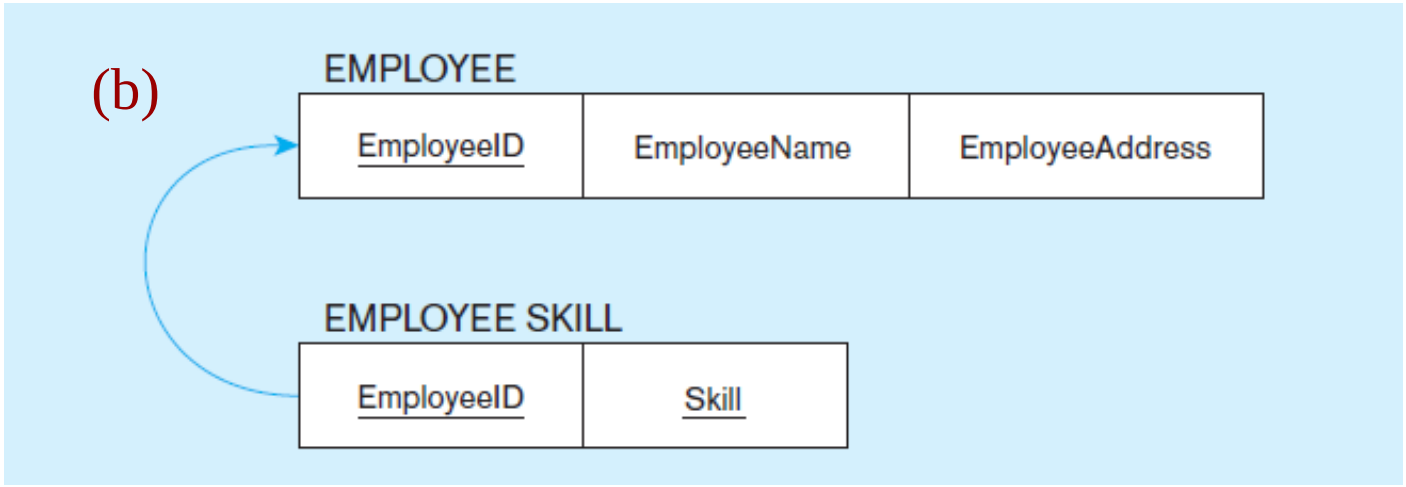
Modeling Relationships

- An entity type is usually modeled as a relation.
- Relationships are usually modeled by having the tuples of one relation point to the tuples of another relation.
- One-to-one, one-to-many, and many-to-one relationships between two entities (relations) can be modeled directly with foreign keys.
- Many-to-many relationships are more of a problem.
  - Must introduce a new relation to model the many-to-many relationship.
  - This relation has foreign keys pointing to tuples in the associated relations (entities).

Figure 4-10 Mapping an entity with a multivalued attribute



Multivalued attribute becomes a separate relation with foreign key



One–to–many relationship between original entity and new relation

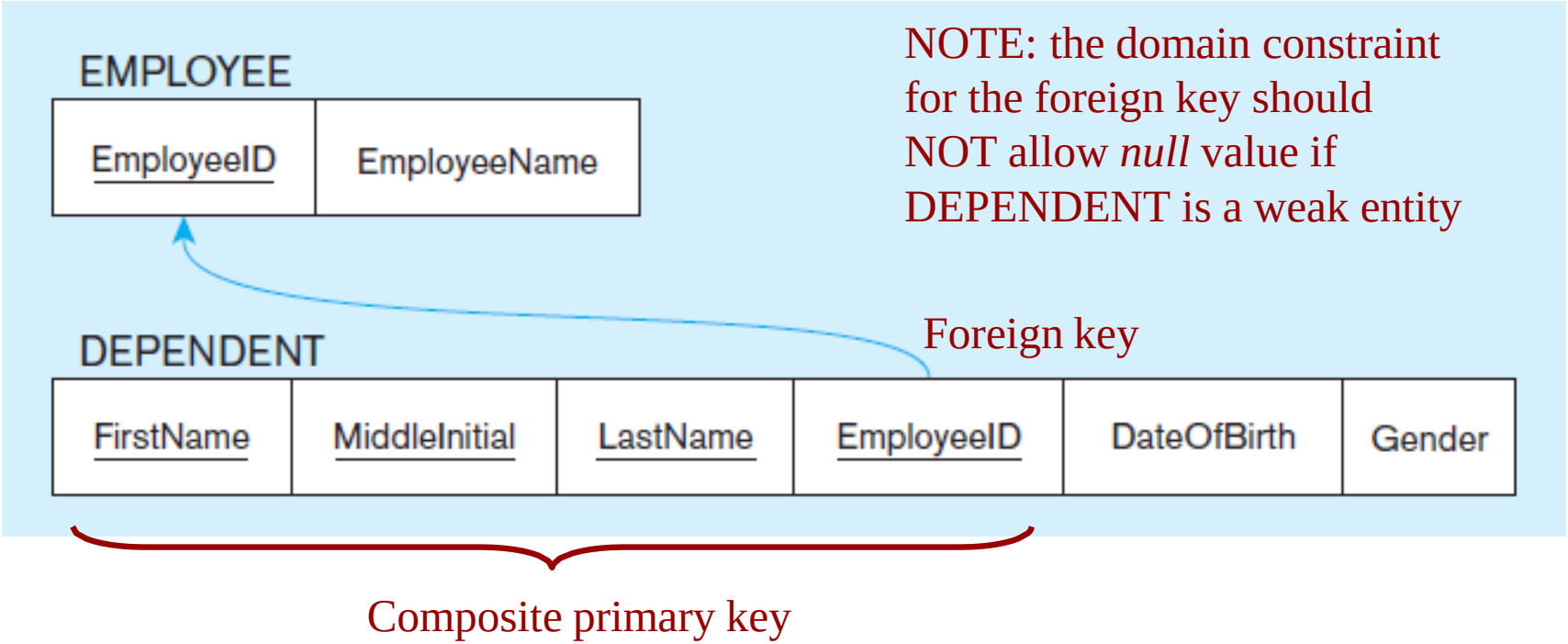
TRANSFORMING EER DIAGRAMS INTO RELATIONS (CONT.)

Mapping Weak Entities

- Becomes a separate relation with a foreign key taken from the superior entity
- Primary key composed of:
  - Partial identifier of weak entity
  - Primary key of identifying relation (strong entity)

Figure 4-11 Example of mapping a weak entity

b) Relations resulting from weak entity

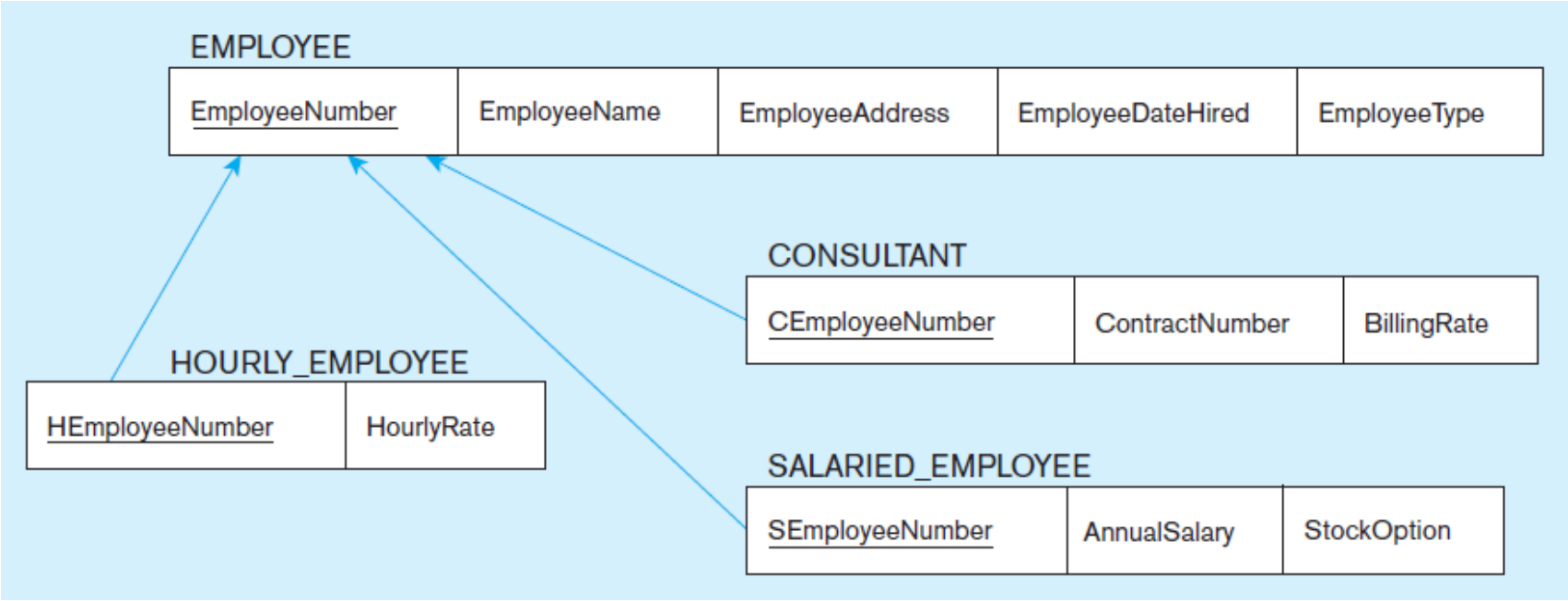


Transforming EER into Relations

Mapping Supertype/Subtype Relationships

- One relation for supertype and for each subtype
- Supertype attributes (including identifier and subtype discriminator) go into supertype relation
- Subtype attributes go into each subtype; primary key of supertype relation also becomes primary key of subtype relation
- 1:1 relationship established between supertype and each subtype, with supertype as primary table

Mapping supertype/subtype relationships to relations



These are implemented as one-to-one relationships.

TABLE 4-2 Summary of EER to Relational Transformations

EER Structure	Relational Representation (Sample Figure)
Regular entity	Create a relation with primary key and nonkey attributes (Figure 4-8)
Composite attribute	Each component of a composite attribute becomes a separate attribute in the target relation (Figure 4-9)
Multivalued attribute	Create a separate relation for multivalued attribute with composite primary key, including the primary key of the entity (Figure 4-10)
Weak entity	Create a relation with a composite primary key (which includes the primary key of the entity on which this entity depends) and nonkey attributes (Figure 4-11)
Binary or unary 1:*N* relationship	Place the primary key of the entity on the one side of the relationship as a foreign key in the relation for the entity on the many side (Figure 4-12; Figure 4-17 for unary relationship)
Binary or unary *M:N* relationship or associative entity without its own key	Create a relation with a composite primary key using the primary keys of the related entities plus any nonkey attributes of the relationship or associative entity (Figure 4-13; Figure 4-15 for associative entity; Figure 4-18 for unary relationship)
Binary or unary 1:1 relationship	Place the primary key of either entity in the relation for the other entity; if one side of the relationship is optional, place the foreign key of the entity on the mandatory side in the relation for the entity on the optional side (Figure 4-14)
Binary or unary *M:N* relationship or associative entity with its own key	Create a relation with the primary key associated with the associative entity plus any nonkey attributes of the associative entity and the primary keys of the related entities as foreign keys (Figure 4-16)
Supertype/subtype relationship	Create a relation for the superclass, which contains the primary and all nonkey attributes in common with all subclasses, plus create a separate relation for each subclass with the same primary key (with the same or local name) but with only the nonkey attributes related to that subclass (Figure 4-20 and 4-21)

DATA NORMALIZATION

Primarily a tool to validate and improve a logical design so that it satisfies certain constraints that *avoid* ***unnecessary duplication of data***

The process of decomposing relations with anomalies to produce smaller, *well-structured* relations

WELL-STRUCTURED RELATIONS

A relation that contains minimal data redundancy and allows users to insert, delete, and update rows without causing data inconsistencies

Goal is to avoid anomalies

- Insertion Anomaly–adding new rows forces user to create duplicate data
- Deletion Anomaly–deleting rows may cause a loss of data that would be needed for other future rows
- Modification Anomaly–changing data in a row forces changes to other rows because of duplication

General rule of thumb: A table should not pertain to more than one entity type.

EXAMPLE

EMPLOYEE2

EmpID	Name	DeptName	Salary	CourseTitle	DateCompleted
100	Margaret Simpson	Marketing	48,000	SPSS	6/19/2015
100	Margaret Simpson	Marketing	48,000	Surveys	10/7/2015
140	Alan Beeton	Accounting	52,000	Tax Acc	12/8/2015
110	Chris Lucero	Info Systems	43,000	Visual Basic	1/12/2015
110	Chris Lucero	Info Systems	43,000	C++	4/22/2015
190	Lorenzo Davis	Finance	55,000		
150	Susan Martin	Marketing	42,000	SPSS	6/19/2015
150	Susan Martin	Marketing	42,000	Java	8/12/2015

Question–Is this a relation?

Answer–Yes: Unique rows and no multivalued attributes

Question–What's the primary key?

Answer–Composite: EmpID, CourseTitle

ANOMALIES IN THIS TABLE

Insertion–can't enter a new employee without having the employee take a class (or at least empty fields of class information)

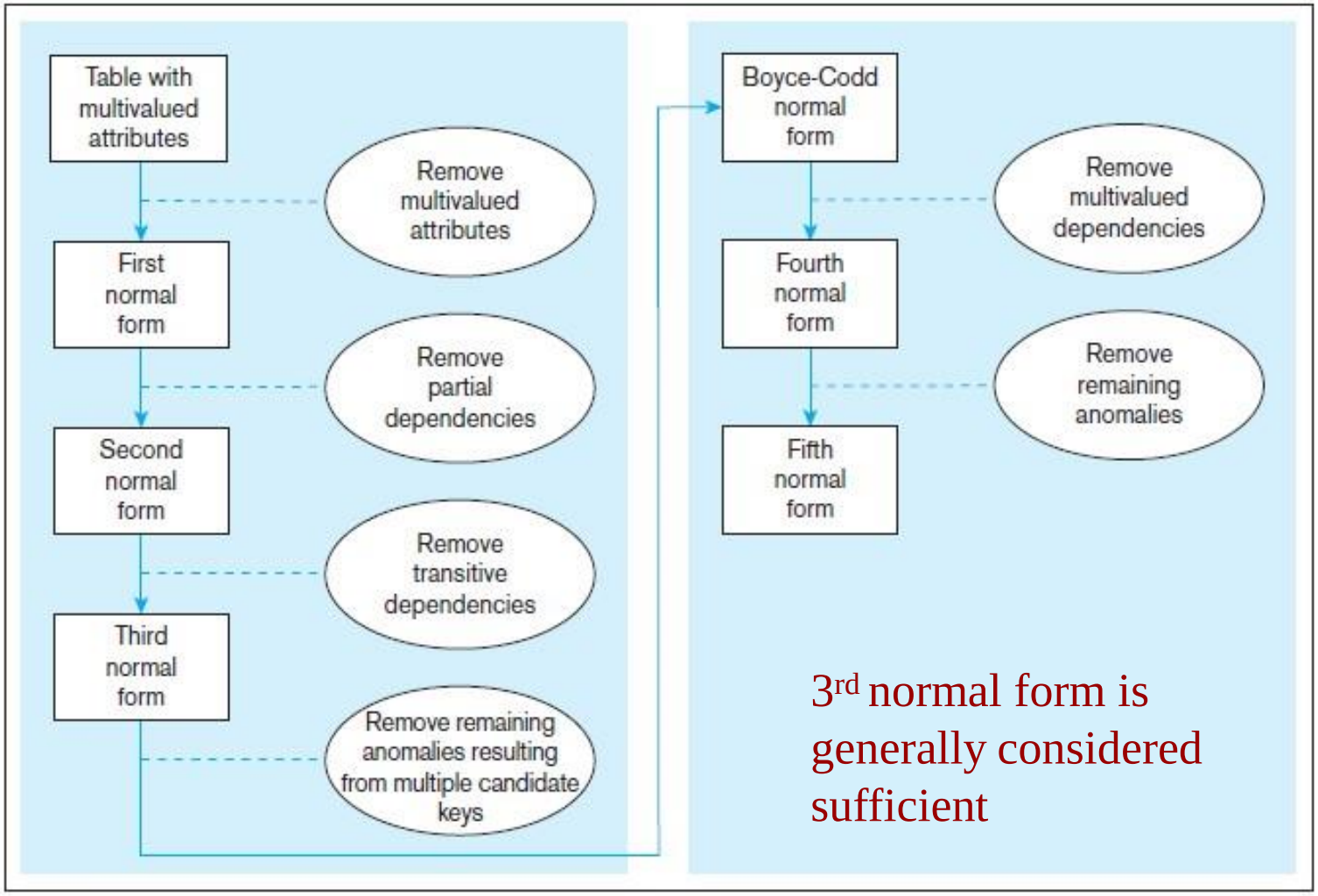
Deletion–if we remove employee 140, we lose information about the existence of a Tax Acc class

Modification–giving a salary increase to employee 100 forces us to update multiple records

Why do these anomalies exist?

Because there are two themes (entity types) in this one relation. This results in data duplication and an unnecessary dependency between the entities.

Figure 4.22 Steps in normalization



FUNCTIONAL DEPENDENCIES AND KEYS

Functional Dependency: The value of one attribute (the *determinant*) determines the value of another attribute

Candidate Key:

- A unique identifier. One of the candidate keys will become the primary key
  - E.g., perhaps there is both credit card number and SS# in a table…in this case both are candidate keys.
- Each non-key field is functionally dependent on every candidate key.

FIRST NORMAL FORM

No multivalued attributes

Every attribute value is atomic

.

Table with multivalued attributes, not in 1st normal form

FIGURE 4-25 INVOICE data (Pine Valley Furniture Company)

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2015	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
					5	Writer's Desk	Cherry	325.00	2
					4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
					4	Entertainment Center	Natural Maple	650.00	3

Note: This is NOT a relation.

Table with no multivalued attributes and unique rows, in 1st normal form

FIGURE 4-26 INVOICE relation (1NF) (Pine Valley Furniture Company)

OrderID	Order Date	Customer ID	Customer Name	Customer Address	ProductID	Product Description	Product Finish	Product StandardPrice	Ordered Quantity
1006	10/24/2015	2	Value Furniture	Plano, TX	7	Dining Table	Natural Ash	800.00	2
1006	10/24/2015	2	Value Furniture	Plano, TX	5	Writer's Desk	Cherry	325.00	2
1006	10/24/2015	2	Value Furniture	Plano, TX	4	Entertainment Center	Natural Maple	650.00	1
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	11	4–Dr Dresser	Oak	500.00	4
1007	10/25/2015	6	Furniture Gallery	Boulder, CO	4	Entertainment Center	Natural Maple	650.00	3

Note: This is a relation, but not a well-structured one.

ANOMALIES IN THIS TABLE

Insertion–if new product is ordered for order 1007 of existing customer, customer data must be re-entered, causing duplication

Deletion–if we delete the Dining Table from Order 1006, we lose information concerning this item's finish and price

Update–changing the price of product ID 4 requires update in multiple records

Why do these anomalies exist?

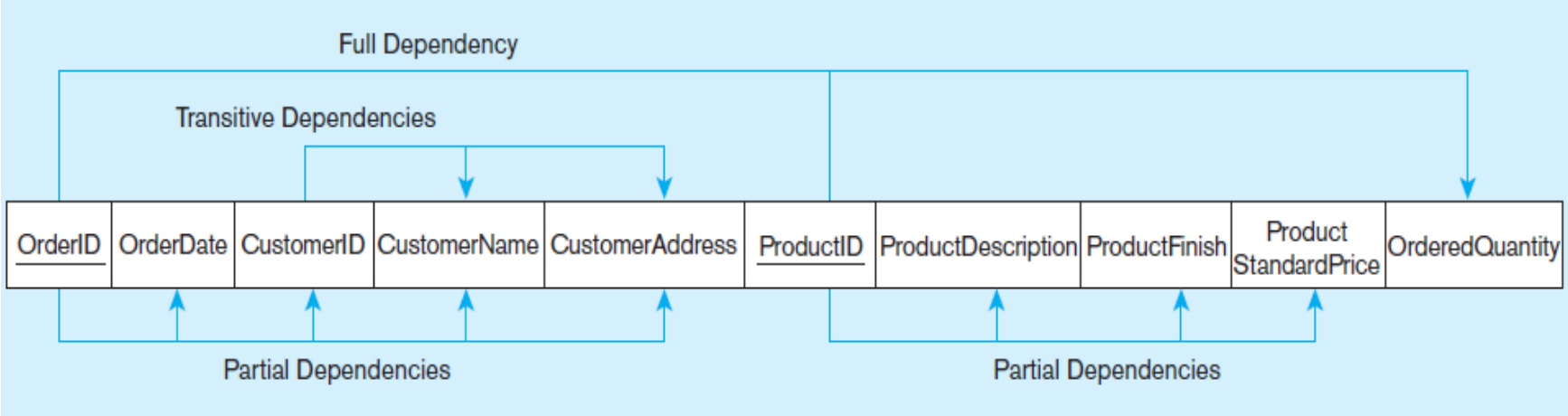
Because there are multiple themes (entity types) in one relation. This results in duplication and an unnecessary dependency between the entities.

SECOND NORMAL FORM

1NF PLUS *every non-key attribute is fully functionally dependent on the ENTIRE primary key*

- Every non-key attribute must be defined by the entire key, not by only part of the key
- No partial functional dependencies

Figure 4-27 Functional dependency diagram for INVOICE



OrderID OrderDate, CustomerID, CustomerName, CustomerAddress

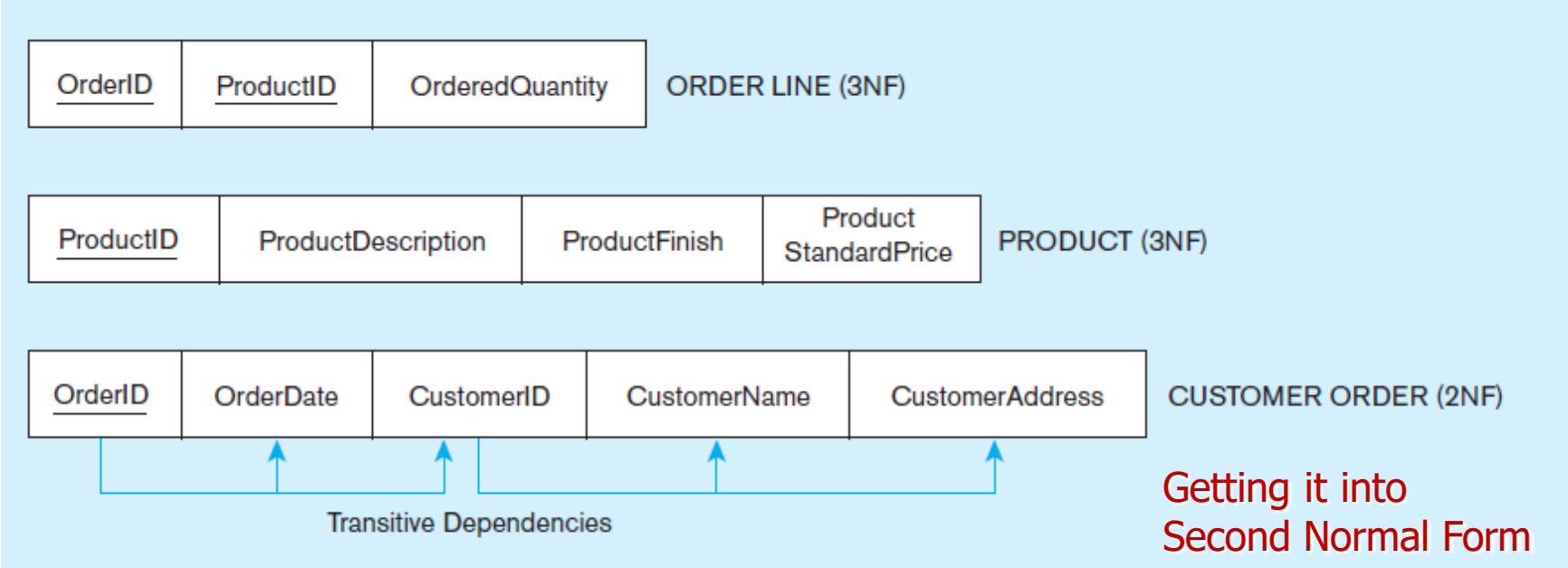
CustomerID CustomerName, CustomerAddress

ProductID ProductDescription, ProductFinish, ProductStandardPrice

OrderID, ProductID OrderQuantity

Therefore, NOT in 2nd Normal Form

Figure 4-28 Removing partial dependencies

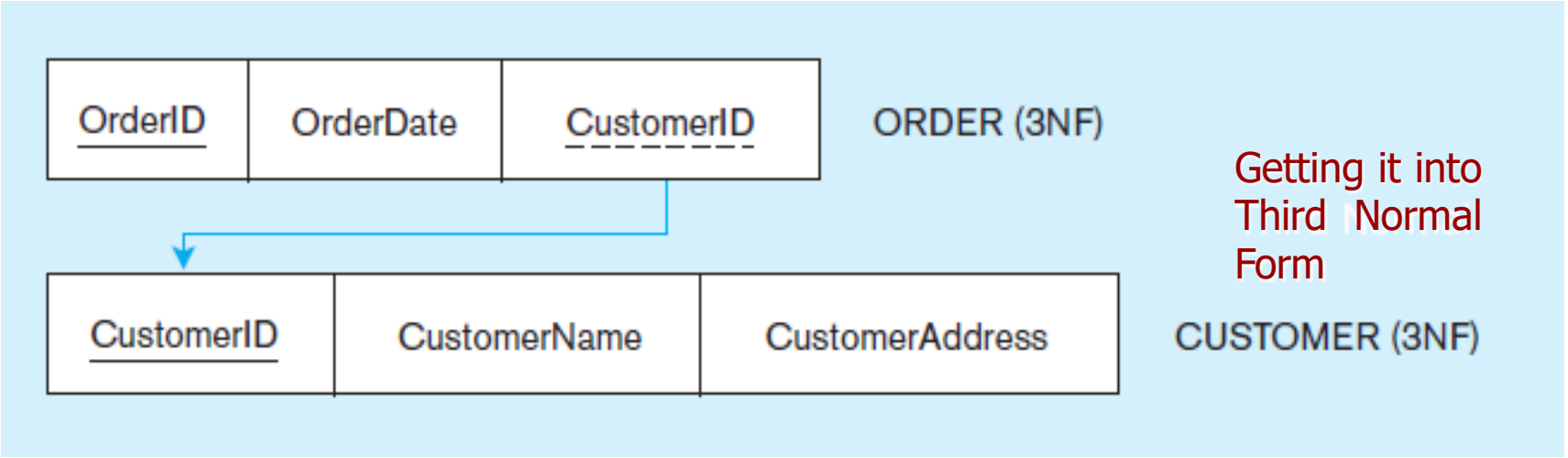


Partial dependencies are removed, but there are still transitive dependencies

THIRD NORMAL FORM

- 2NF PLUS *no transitive dependencies* (functional dependencies on non-primary-key attributes)
- Note: This is called transitive, because the primary key is a determinant for another attribute, which in turn is a determinant for a third
- Solution: Non-key determinant with transitive dependencies go into a new table; non-key determinant becomes primary key in the new table and stays as foreign key in the old table

Figure 4-29 Removing partial dependencies



Transitive dependencies are removed.

Joins

- Join–a relational operation that causes two or more tables with a common domain to be combined into a single table or view

- FROM clause of the query specifies the table or tables from which the data is to be retrieved
- Inner joins return only rows from the tables that match on a common value
- Outer joins return the same matched rows as the inner join, plus unmatched rows from one table or the other

Joining Tables

- JOIN USING syntax
  - Returns only the rows with matching values in the column indicated in the USING clause— and that column must exist in both tables

SELECT *column-list* **FROM** *table1* **JOIN** *table2* **USING** (*common-column*)

- JOIN ON syntax
  - Express a join when the tables have no common attribute names
  - Query returns only the rows that meet the indicated join condition

SELECT *column-list* **FROM** *table1* **JOIN** *table2* **ON** join-condition

PRODUCT & VENDOR Tables

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

V_CODE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
21225	Bryson, Inc.	Smithson	615	223-3234	TN	Y
21226	SuperLoo, Inc.	Flushing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	901	678-3998	GA	Y
24004	Brackman Bros.	Browning	615	228-1410	TN	N
24288	ORDVA, Inc.	Hakford	615	898-1234	TN	Y
25443	B&K, Inc.	Smith	904	227-0093	FL	N
25501	Damal Supplies	Smythe	615	890-3529	TN	N
25595	Rubicon Systems	Orton	904	456-0092	FL	Y

JOIN USING Example

```
SELECT
    P_CODE,
    P_DESCRIPT,
    V_CODE,  V_NAME,
    V_AREACODE,
    V_PHONE
FROM
    PRODUCT
```

P_CODE	P_DESCRIPT	V_CODE	V_NAME	V_AREACODE	V_PHONE
23109-HB	Claw hammer	21225	Bryson, Inc.	615	223-3234
SM-18277	1.25-in. metal screw, 25	21225	Bryson, Inc.	615	223-3234
SW-23116	2.5-in. wd. screw, 50	21231	D&E Supply	615	228-3245
13-Q2/P2	7.25-in. pwr. saw blade	21344	Gomez Bros.	615	889-2546
14-Q1/L3	9.00-in. pwr. saw blade	21344	Gomez Bros.	615	889-2546
54778-2T	Rat-tail file, 1/8-in. fine	21344	Gomez Bros.	615	889-2546
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	23119	Randsets Ltd.	901	678-3998
1558-QW1	Hrd. cloth, 1/2-in., 3x50	23119	Randsets Ltd.	901	678-3998
2232/QTY	B&D jigsaw, 12-in. blade	24288	ORDVA, Inc.	615	898-1234
2232/QWE	B&D jigsaw, 8-in. blade	24288	ORDVA, Inc.	615	898-1234
89-WRE-Q	Hicut chain saw, 16 in.	24288	ORDVA, Inc.	615	898-1234
11QER/31	Power painter, 15 psi., 3-nozzle	25595	Rubicon Systems	904	456-0092
2238/QPD	B&D cordless drill, 1/2-in.	25595	Rubicon Systems	904	456-0092
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	25595	Rubicon Systems	904	456-0092

```
    JOIN
  VENDOR USING (V_CODE);
```

JOIN ON Example

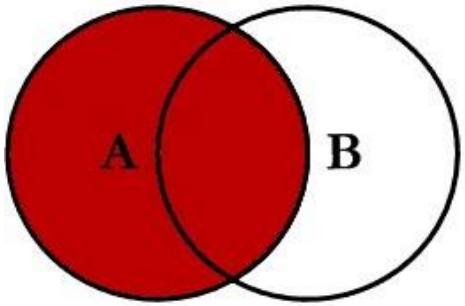
```
SELECT
    INVOICE.INV_NUMBER,
    PRODUCT.P_CODE,
    P_DESCRIPT,
    LINE_UNITS,
    LINE_PRICE,
    P_QOH*P_PRICE
FROM

    INVOICE
        JOIN
    LINE ON INVOICE.INV_NUMBER = LINE.INV_NUMBER
        JOIN
    PRODUCT ON LINE.P_CODE = PRODUCT.P_CODE;
```

INV_NUMBER	P_CODE	P_DESCRIPT	LINE_UNITS	LINE_PRICE	P_QOH*P_PRICE
1001	13-Q2/P2	7.25-in. pwr. saw blade	1.00	14.99	479.68
1001	23109-HB	Claw hammer	1.00	9.95	228.85
1002	54778-2T	Rat-tail file, 1/8-in. fine	2.00	4.99	214.57
1003	2238/QPD	B&D cordless drill, 1/2-in.	1.00	38.95	467.40
1003	1546-QQ2	Hrd. cloth, 1/4-in., 2x50	1.00	39.95	599.25
1003	13-Q2/P2	7.25-in. pwr. saw blade	5.00	14.99	479.68
1004	54778-2T	Rat-tail file, 1/8-in. fine	3.00	4.99	214.57
1004	23109-HB	Claw hammer	2.00	9.95	228.85
1005	PVC23DRT	PVC pipe, 3.5-in., 8-ft	12.00	5.87	1103.56
1006	SM-18277	1.25-in. metal screw, 25	3.00	6.99	1202.28
1006	2232/QTY	B&D jigsaw, 12-in. blade	1.00	109.92	879.36
1006	23109-HB	Claw hammer	1.00	9.95	228.85
1006	89-WRE-Q	Hicut chain saw, 16 in.	1.00	256.99	2826.89
1007	13-Q2/P2	7.25-in. pwr. saw blade	2.00	14.99	479.68
1007	54778-2T	Rat-tail file, 1/8-in. fine	1.00	4.99	214.57
1008	PVC23DRT	PVC pipe, 3.5-in., 8-ft	5.00	5.87	1103.56
1008	WR3/TT3	Steel matting, 4'x8'x1/6", .5" mesh	3.00	119.95	2159.10
1008	23109-HB	Claw hammer	1.00	9.95	228.85

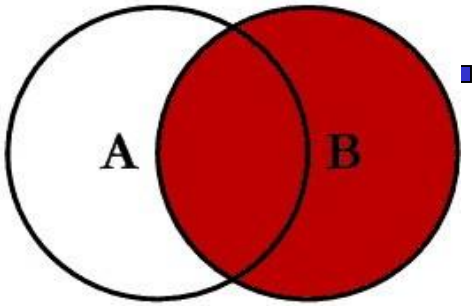
PROCESSING MULTIPLE TABLES

- Outer join–a join in which rows that do not have matching values in common columns are nonetheless included in the result table (as opposed to *inner* join, in which rows must have matching values in order to appear in the result table)
- Joining tables with an alias
  - An alias may be used to identify the source table from which the data is taken
  - The ability to specify a table alias is very useful

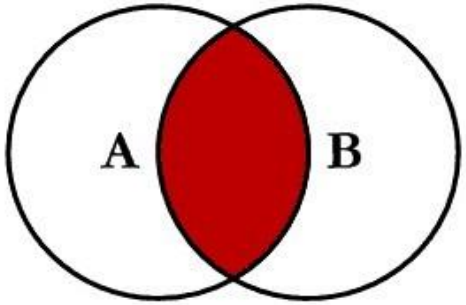
SQL JOINS



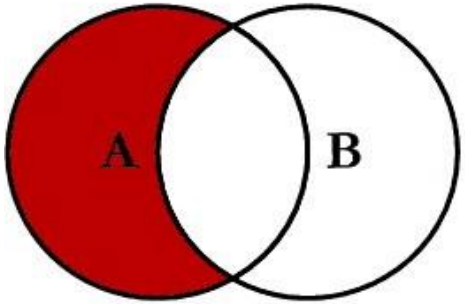

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



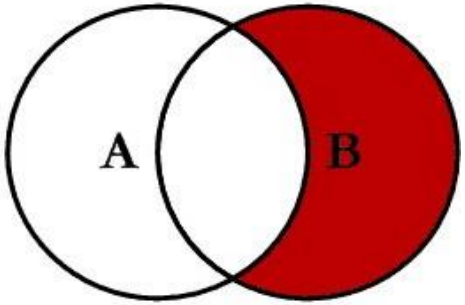

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



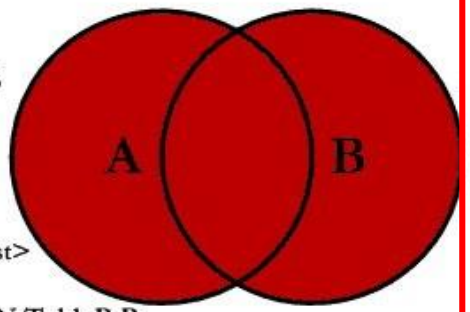

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



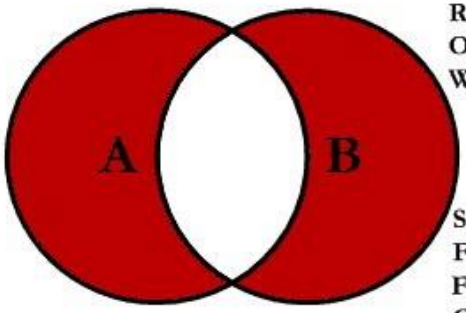

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



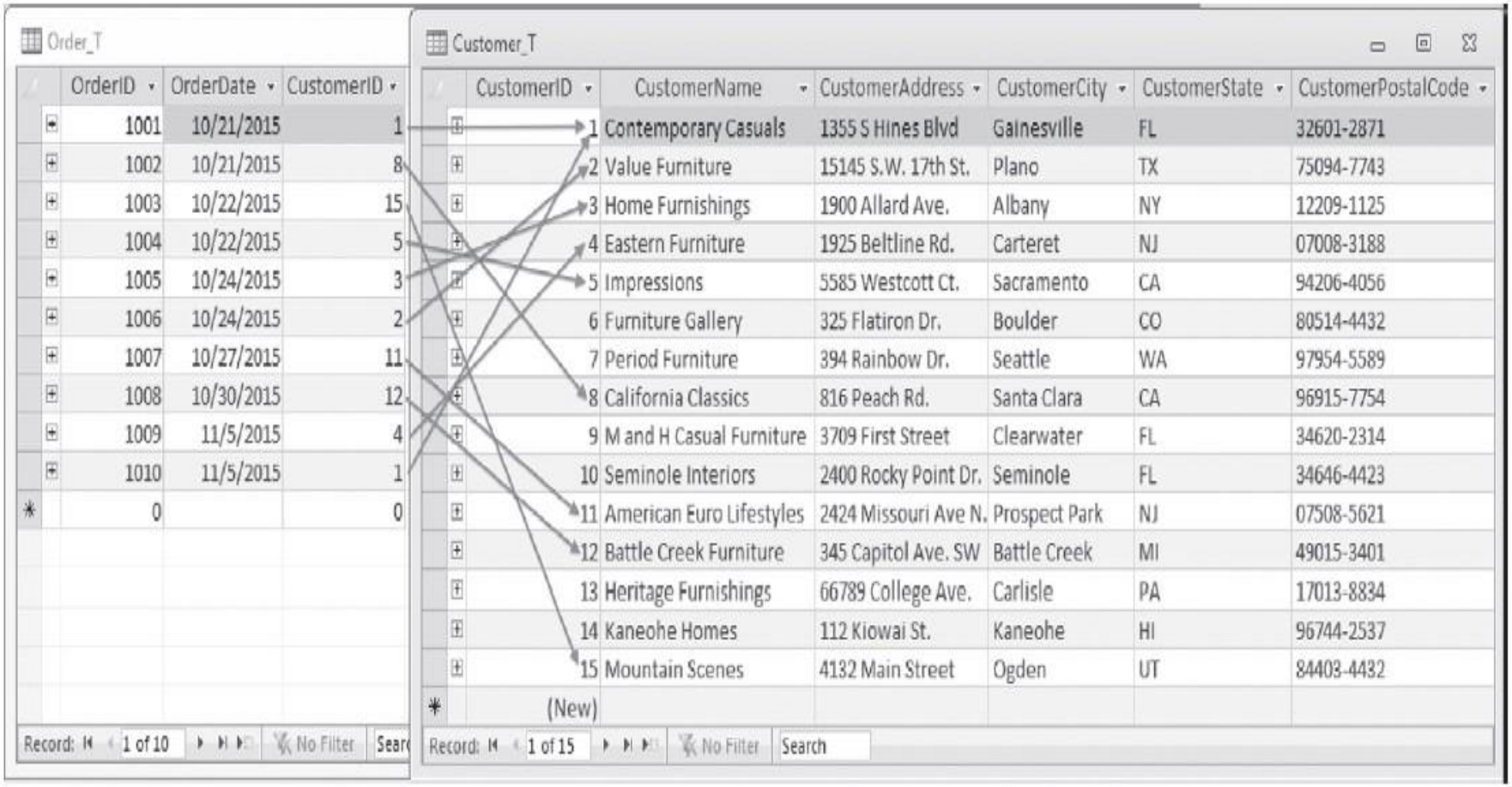

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
```




```
SELECT <select_list>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
OR B.Key IS NULL
```



Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2015	1
1002	10/21/2015	8
1003	10/22/2015	15
1004	10/22/2015	5
1005	10/24/2015	3
1006	10/24/2015	2
1007	10/27/2015	11
1008	10/30/2015	12
1009	11/5/2015	4
1010	11/5/2015	1
0		0

Record: 1 of 10 No Filter

Customer_T

CustomerID	CustomerName	CustomerAddress	CustomerCity	CustomerState	CustomerPostalCode
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589
8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
9	M and H Casual Furniture	3709 First Street	Clearwater	FL	34620-2314
10	Seminole Interiors	2400 Rocky Point Dr.	Seminole	FL	34646-4423
11	American Euro Lifestyles	2424 Missouri Ave N.	Prospect Park	NJ	07508-5621
12	Battle Creek Furniture	345 Capitol Ave. SW	Battle Creek	MI	49015-3401
13	Heritage Furnishings	66789 College Ave.	Carlisle	PA	17013-8834
14	Kaneohe Homes	112 Kiowai St.	Kaneohe	HI	96744-2537
15	Mountain Scenes	4132 Main Street	Ogden	UT	84403-4432
(New)					

Record: 1 of 15 No Filter Search

These tables are used in queries that follow

EQUI-JOIN EXAMPLE

For each customer who placed an order, what is the customer's name and order number?

```
SELECT Customer_T.CustomerID, Order_T.CustomerID,
   CustomerName, OrderID
     FROM Customer_T, Order_T
       WHERE Customer_T.CustomerID = Order_T. CustomerID
       ORDER BY OrderID
```

Result:

CUSTOMERID	CUSTOMERID	CUSTOMERNAME	ORDERID
1	1	Contemporary Casuals	1001
8	8	California Classics	1002
15	15	Mountain Scenes	1003
5	5	Impressions	1004
3	3	Home Furnishings	1005
2	2	Value Furniture	1006
11	11	American Euro Lifestyles	1007
12	12	Battle Creek Furniture	1008
4	4	Eastern Furniture	1009
1	1	Contemporary Casuals	1010

10 rows selected.

Customer ID appears twice in the result

An INNER JOIN operation

```
SELECT * FROM table1
INNER JOIN
table2 ON table1.room_number = table2.room_number
```

NATURAL JOIN EXAMPLE

- For each customer who placed an order, what is the customer's name and order number?

Join involves multiple tables in FROM clause

```
SELECT Customer_T.CustomerID, CustomerName, OrderID
FROM Customer_T NATURAL JOIN Order_T ON
Customer_T.CustomerID = Order_T.CustomerID;
```

ON clause performs the equality check for common columns of the two tables

CUSTOMERID	CUSTOMERNAME	ORDERID
1	Contemporary Casuals	1001
8	California Classics	1002
15	Mountain Scenes	1003
5	Impressions	1004
3	Home Furnishings	1005
2	Value Furniture	1006
11	American Euro Lifestyles	1007
12	Battle Creek Furniture	1008
4	Eastern Furniture	1009
1	Contemporary Casuals	1010

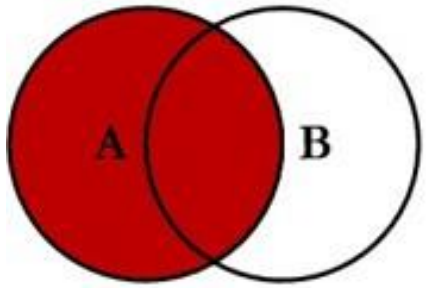
A NATURAL JOIN is **a JOIN operation**

OUTER JOIN EXAMPLE

- List the customer name, ID number, and order number for all customers. Include customer information even for customers that do have an order.

```
SELECT Customer_T.CustomerID, CustomerName, OrderID
   FROM Customer_T LEFT OUTER JOIN Order_T
   WHERE Customer_T.CustomerID = Order_T. CustomerID;
```

LEFT OUTER JOIN clause causes customer data to appear even if there is no corresponding order data



There really is no difference between a LEFT JOIN and a LEFT OUTER JOIN,
(RIGHT JOIN or RIGHT OUTER JOIN)

Outer Join Results

CUSTOMERID	CUSTOMERNAME	ORDERID
1	Contemporary Casuals	1001
1	Contemporary Casuals	1010
2	Value Furniture	1006
3	Home Furnishings	1005
4	Eastern Furniture	1009
5	Impressions	1004
6	Furniture Gallery	
7	Period Furniture	
8	California Classics	1002
9	M & H Casual Furniture	
10	Seminole Interiors	
11	American Euro Lifestyles	1007
12	Battle Creek Furniture	1008
13	Heritage Furnishings	
14	Kaneohe Homes	
15	Mountain Scenes	1003

16 rows selected.

MULTIPLE TABLE JOIN EXAMPLE

- Assemble all information necessary to create an invoice for order number 1006

```
SELECT Customer_T.CustomerID, CustomerName, CustomerAddress,
   CustomerCity, CustomerState, CustomerPostalCode, Order_T.OrderID,
   OrderDate, OrderedQuantity, ProductDescription, StandardPrice,
   (OrderedQuantity * ProductStandardPrice)
FROM Customer_T, Order_T, OrderLine_T, Product_T
  WHERE Order_T.CustomerID = Customer_T.CustomerID
    AND Order_T.OrderID = OrderLine_T.OrderID
    AND OrderLine_T.ProductID = Product_T.ProductID
    AND Order_T.OrderID = 1006;
```

Four tables involved in this join

Each pair of tables requires an equality-check condition in the WHERE clause, matching primary keys against foreign keys.

SELF-JOIN EXAMPLE

Query: What are the employee ID and name of each employee and the name of his or her supervisor (label the supervisor's name Manager)?

```
SELECT E.EmployeeID, E.EmployeeName, M.EmployeeName AS Manager
  FROM Employee_T E, Employee_T M
  WHERE E.EmployeeSupervisor = M.EmployeeID;
```

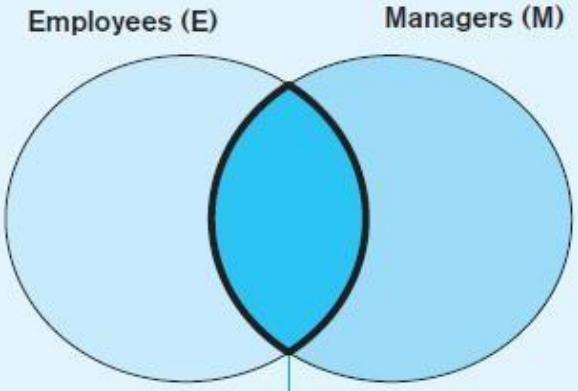
The same table is used on both sides of the join; distinguished using table aliases

Result:

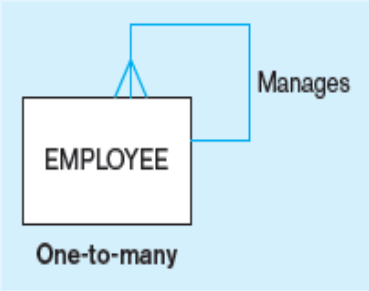
EMPLOYEEID	EMPLOYEENAME	MANAGER
123-44-347	Jim Jason	Robert Lewis

Self-joins are usually used on tables with unary relationships.

Figure 7-5 Example of a self-join



Employees who have supervisors; i.e.,
WHERE E.EmployeeSupervisor = M.EmployeeID



Employees (E)

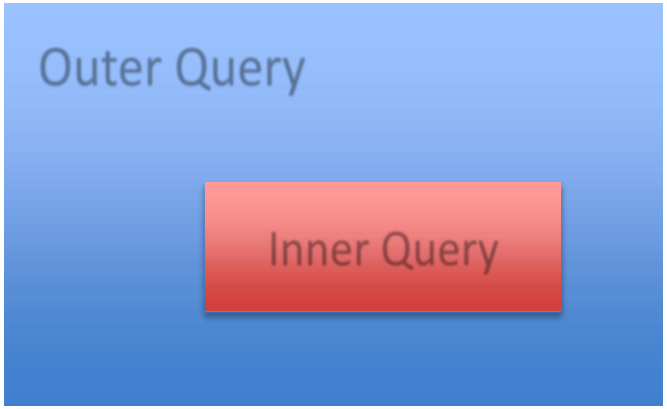
EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

Managers (M)

EmployeeID	EmployeeName	EmployeeSupervisor
098-23-456	Sue Miller	
107-55-789	Stan Getz	
123-44-347	Jim Jason	678-44-546
547-33-243	Bill Blass	
678-44-546	Robert Lewis	

Subqueries

- A subquery is a query (SELECT statement) inside another query
  - A subquery is normally expressed inside parentheses
  - The output of an inner query is used as the input for the outer query
  - The entire SQL statement is sometimes referred to as a nested query




```
SELECT
    P_CODE,
    P_PRICE
FROM
    PRODUCT
WHERE
    P_PRICE >= (SELECT
                    AVG(P_PRICE)
                FROM
                    PRODUCT);
```

- Subquery can return one or more values: single value, list of values etc

Subqueries

- WHERE subqueries
- IN subqueries
- HAVING subqueries
- FROM subqueries
- EXIST subqueries
- Attribute list subqueries
- Subqueries involving ALL and ANY operators

PRODUCT & VENDOR Tables

P_CODE	P_DESCRIPT	P_INDATE	P_QOH	P_MIN	P_PRICE	P_DISCOUNT	V_CODE
11QER/31	Power painter, 15 psi., 3-nozzle	2017-11-03	8	5	109.99	0.00	25595
13-Q2/P2	7.25-in. pwr. saw blade	2017-12-13	32	15	14.99	0.05	21344
14-Q1/L3	9.00-in. pwr. saw blade	2017-11-13	18	12	17.49	0.00	21344
1546-QQ2	Hrd. cloth, 1/4-in., 2x50	2018-01-15	15	8	39.95	0.00	23119
1558-QW1	Hrd. cloth, 1/2-in., 3x50	2018-01-15	23	5	43.99	0.00	23119
2232/QTY	B&D jigsaw, 12-in. blade	2017-12-30	8	5	109.92	0.05	24288
2232/QWE	B&D jigsaw, 8-in. blade	2017-12-24	6	5	99.87	0.05	24288
2238/QPD	B&D cordless drill, 1/2-in.	2018-01-20	12	5	38.95	0.05	25595
23109-HB	Claw hammer	2018-01-20	23	10	9.95	0.10	21225
23114-AA	Sledge hammer, 12 lb.	2018-01-02	8	5	14.40	0.05	NULL
54778-2T	Rat-tail file, 1/8-in. fine	2017-12-15	43	20	4.99	0.00	21344
89-WRE-Q	Hicut chain saw, 16 in.	2018-02-07	11	5	256.99	0.05	24288
PVC23DRT	PVC pipe, 3.5-in., 8-ft	2018-02-20	188	75	5.87	0.00	NULL
SM-18277	1.25-in. metal screw, 25	2018-03-01	172	75	6.99	0.00	21225
SW-23116	2.5-in. wd. screw, 50	2018-02-24	237	100	8.45	0.00	21231
WR3/TT3	Steel matting, 4'x8'x1/6", .5" m...	2018-01-17	18	5	119.95	0.10	25595

V_CODE	V_NAME	V_CONTACT	V_AREACODE	V_PHONE	V_STATE	V_ORDER
21225	Bryson, Inc.	Smithson	615	223-3234	TN	Y
21226	SuperLoo, Inc.	Flushing	904	215-8995	FL	N
21231	D&E Supply	Singh	615	228-3245	TN	Y
21344	Gomez Bros.	Ortega	615	889-2546	KY	N
22567	Dome Supply	Smith	901	678-1419	GA	N
23119	Randsets Ltd.	Anderson	901	678-3998	GA	Y
24004	Brackman Bros.	Browning	615	228-1410	TN	N
24288	ORDVA, Inc.	Hakford	615	898-1234	TN	Y
25443	B&K, Inc.	Smith	904	227-0093	FL	N
25501	Damal Supplies	Smythe	615	890-3529	TN	N
25595	Rubicon Systems	Orton	904	456-0092	FL	Y

WHERE Subqueries

- Most common subquery: use an inner SELECT subquery on the right side of a WHERE comparison expression

```
SELECT
    P_CODE,
    P_PRICE
FROM
    PRODUCT
WHERE
    P_PRICE >= (SELECT
                    AVG(P_PRIC
                E)   FROM
                    PRODUCT);
```

P_CODE	P_PRICE
11QER/31	109.99
13-Q2/P2	14.99
14-Q1/L3	17.49
1546-QQ2	39.95
1558-QW1	43.99
2232/QTY	109.92
2232/QWE	99.87
2238/QPD	38.95
23109-HB	9.95
23114-AA	14.40
54778-2T	4.99
89-WRE-Q	256.99
PVC23DRT	5.87
SM-18277	6.99
SW-23116	8.45
WR3/TT3	119.95

P_CODE	P_PRICE
11QER/31	109.99
2232/QTY	109.92
2232/QWE	99.87
89-WRE-Q	256.99
WR3/TT3	119.95

IN Subqueries

- Compare a single attribute to a list of values that's not known beforehand, but can be derived using a (sub)query

```
SELECT
    P_CODE,
FROM
    VENDOR
```

P_CODE	P_DESCRIPT	V_NAME	V_CONTACT
13-Q2/P2	7.25-in. pwr. saw blade	Gomez Bros.	Ortega
14-Q1/L3	9.00-in. pwr. saw blade	Gomez Bros.	Ortega
2232/QTY	B&D jigsaw, 12-in. blade	ORDVA, Inc.	Hakford
2232/QWE	B&D jigsaw, 8-in. blade	ORDVA, Inc.	Hakford
23109-HB	Claw hammer	Bryson, Inc.	Smithson
89-WRE-Q	Hicut chain saw, 16 in.	ORDVA, Inc.	Hakford

```
WHERE
    P_CODE IN (SELECT P_CODE FROM PRODUCT
               WHERE P_DESCRIPT LIKE '%hammer%'
               OR P_DESCRIPT LIKE '%saw%');
```

HAVING Subqueries

- HAVING clause: used to restrict the output of a GROUP BY query by applying conditional criteria to the grouped rows

```
SELECT
    P_CODE,
    SUM(LINE_UNITS) AS
    TOTALUNITS
FROM
    LINE
GROUP BY P_CODE
HAVING SUM(LINE_UNITS) >
(SELECT
      AVG(LINE_UNITS)
   FROM
      LINE);
```

Find all products with total units above average across all products

P_CODE	TOTALUNITS
13-Q2/P2	8.00
1546-QQ2	1.00
2232/QTY	1.00
2238/QPD	1.00
23109-HB	5.00
54778-2T	6.00
89-WRE-Q	1.00
PVC23DRT	17.00
SM-18277	3.00
WR3/TT3	3.00

P_CODE	TOTALUNITS
13-Q2/P2	8.00
23109-HB	5.00
54778-2T	6.00
PVC23DRT	17.00
SM-18277	3.00
WR3/TT3	3.00

FROM Subqueries

- FROM clause specifies the table(s) from which the data will be drawn. Can use a subquery in lieu of a table.

Find all customers (name & balance) who have purchased P_CODE 13-Q2/P2

```
SELECT
    CP_INV.CUS_CODE, CP.CUS_LNAME, CP.CUS_FNAME, CP.CUS_BALANCE,
FROM
    (SELECT
        INVOICE.CUS_CODE
    FROM
        INVOICE
    JOIN LINE ON INVOICE.INV_NUMBER = LINE.INV_NUMBER
    WHERE P_CODE = '13-Q2/P2') CP_INV
        JOIN
    CUSTOMER CP ON CP_INV.CUS_CODE = CP.CUS_CODE;
```

CUS_CODE	CUS_LNAME	CUS_FNAME	CUS_BALANCE
10014	Orlando	Myron	0.00
10012	Smith	Kathy	345.86
10015	O'Brian	Amy	0.00

Attribute list subqueries

- Inline subquery in attribute list: subquery expression
  - Example: list the difference between each product's price and the average product price

```
SELECT
        P_CODE,
        P_PRICE,
        P_PRICE - (SELECT
                      AVG(P_PRICE)
                   FROM
                      PRODUCT) AS PRICE_DIFF
FROM
    PRODUCT;
```

P_CODE	P_PRICE	PRICE_DIFF
11QER/31	109.99	53.568750
13-Q2/P2	14.99	-41.431250
14-Q1/L3	17.49	-38.931250
1546-QQ2	39.95	-16.471250
1558-QW1	43.99	-12.431250
2232/QTY	109.92	53.498750
2232/QWE	99.87	43.448750
2238/QPD	38.95	-17.471250
23109-HB	9.95	-46.471250
23114-AA	14.40	-42.021250
54778-2T	4.99	-51.431250
89-WRE-Q	256.99	200.568750
PVC23DRT	5.87	-50.551250
SM-18277	6.99	-49.431250
SW-23116	8.45	-47.971250
WR3/TT3	119.95	63.528750

Subqueries with ANY or ALL Operators

- Multirow subquery operators: ALL and ANY
  - ALL operator compares a single value with a list of values returned by the first subquery using a comparison operator other than equals
  - ANY operator compares a single value to a list of values and select only the rows greater than or less than any value in the list

Subquery with ALL Operator Example

Which products cost more than all the products provided by vendors from Florida?

```
SELECT
    P_CODE,
    P_QOH*P_PRICE AS TOTALVALUE
FROM
    PRODUCT
WHERE
    P_QOH*P_PRICE > ALL(SELECT
                            P_QOH*P_PRICE
                        FROM
                            PRODUCT
                        WHERE
                             V_CODE IN (SELECT V_CODE
                                        FROM VENDOR
                                        WHERE V_STATE = 'FL'));
```

P_CODE	TOTALVALUE
23114-AA	115.20
54778-2T	214.57
23109-HB	228.85
14-Q1/L3	314.82
2238/QPD	467.40
13-Q2/P2	479.68
2232/QWE	599.22
1546-QQ2	599.25
2232/QTY	879.36
11QER/31	879.92
1558-QW1	1011.77
PVC23DRT	1103.56
SM-18277	1202.28
SW-23116	2002.65
WR3/TT3	2159.10
89-WRE-Q	2826.89