Chapter 8:

Advanced SQL

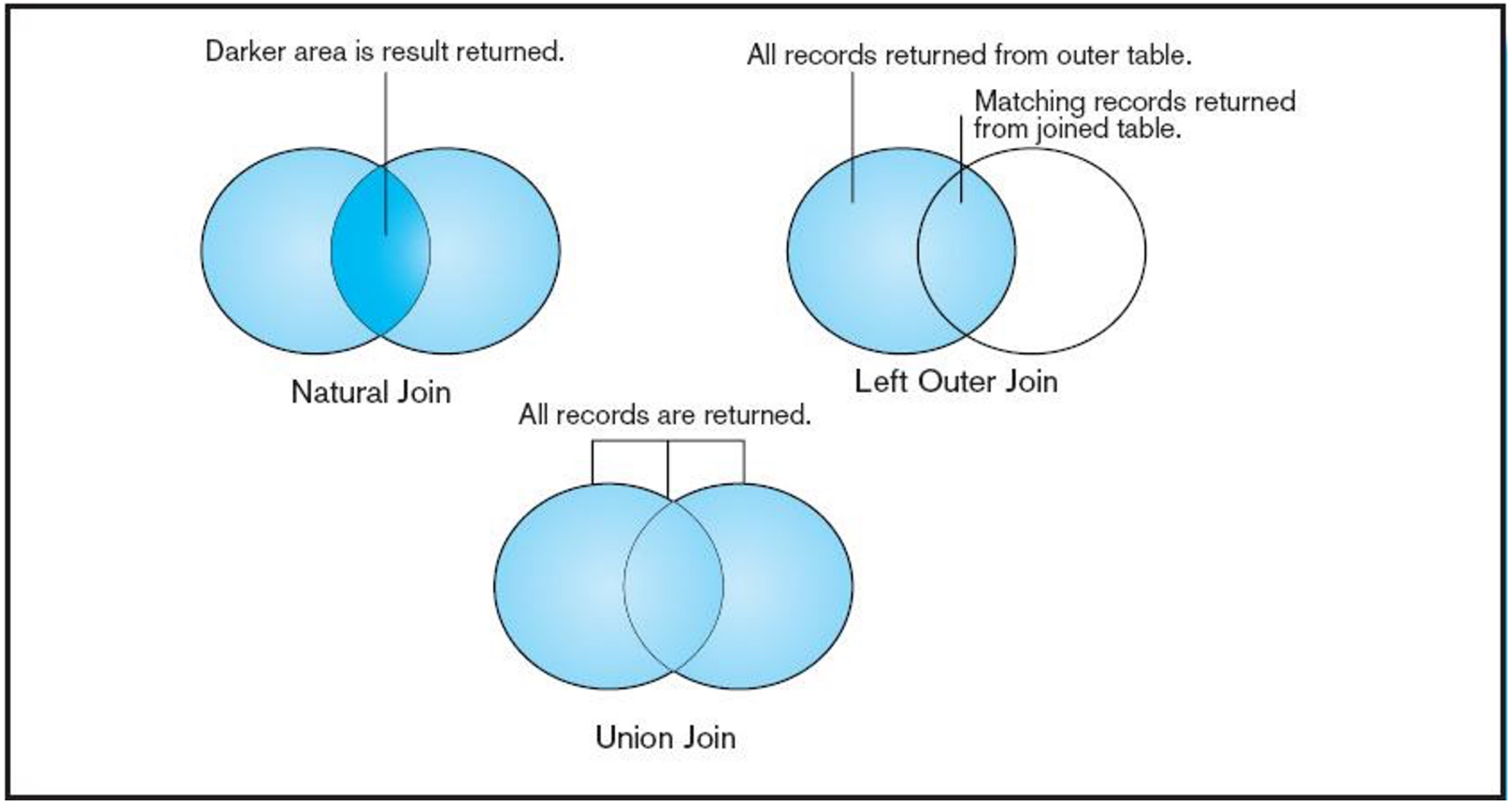
Processing Multiple Tables–Joins

- **Join**–a relational operation that causes two or more tables with a common domain to be combined into a single table or view
- **Equi-join**–a join in which the joining condition is based on equality between values in the common columns; common columns appear redundantly in the result table
- **Natural join**–an equi-join in which one of the duplicate columns is eliminated in the result table
- **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)
- **Union join**–includes all columns from each table in the join, and an instance for each row of each table

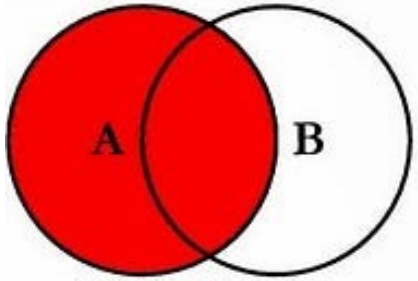
The common columns in joined tables are usually the primary key of the dominant table and the foreign key of the dependent table in 1:M relationships

Figure 8-2

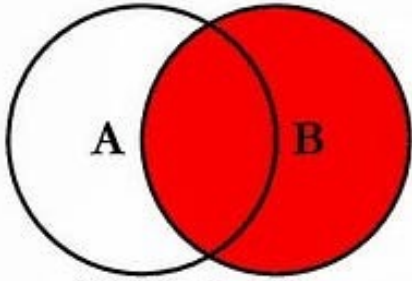
Visualization of different join types with results returned in shaded area



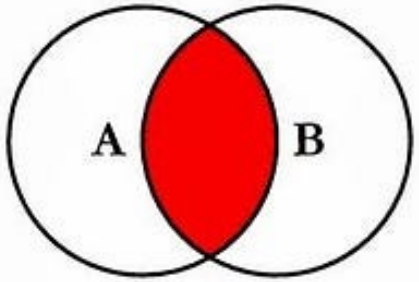
SQL JOINS



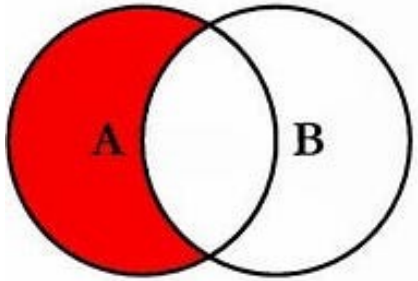

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
```



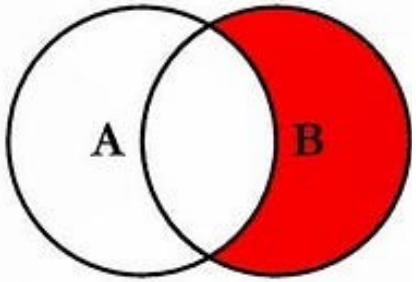

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
```



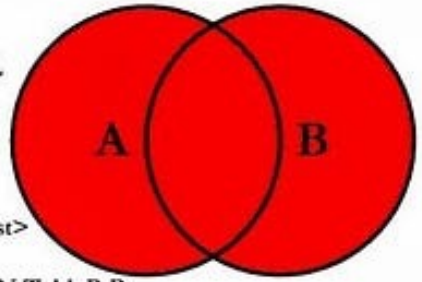

```
SELECT <select_list>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
```



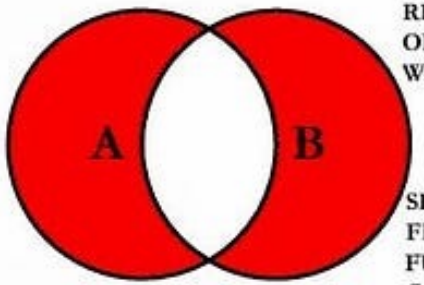

```
SELECT <select_list>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE B.Key IS NULL
```



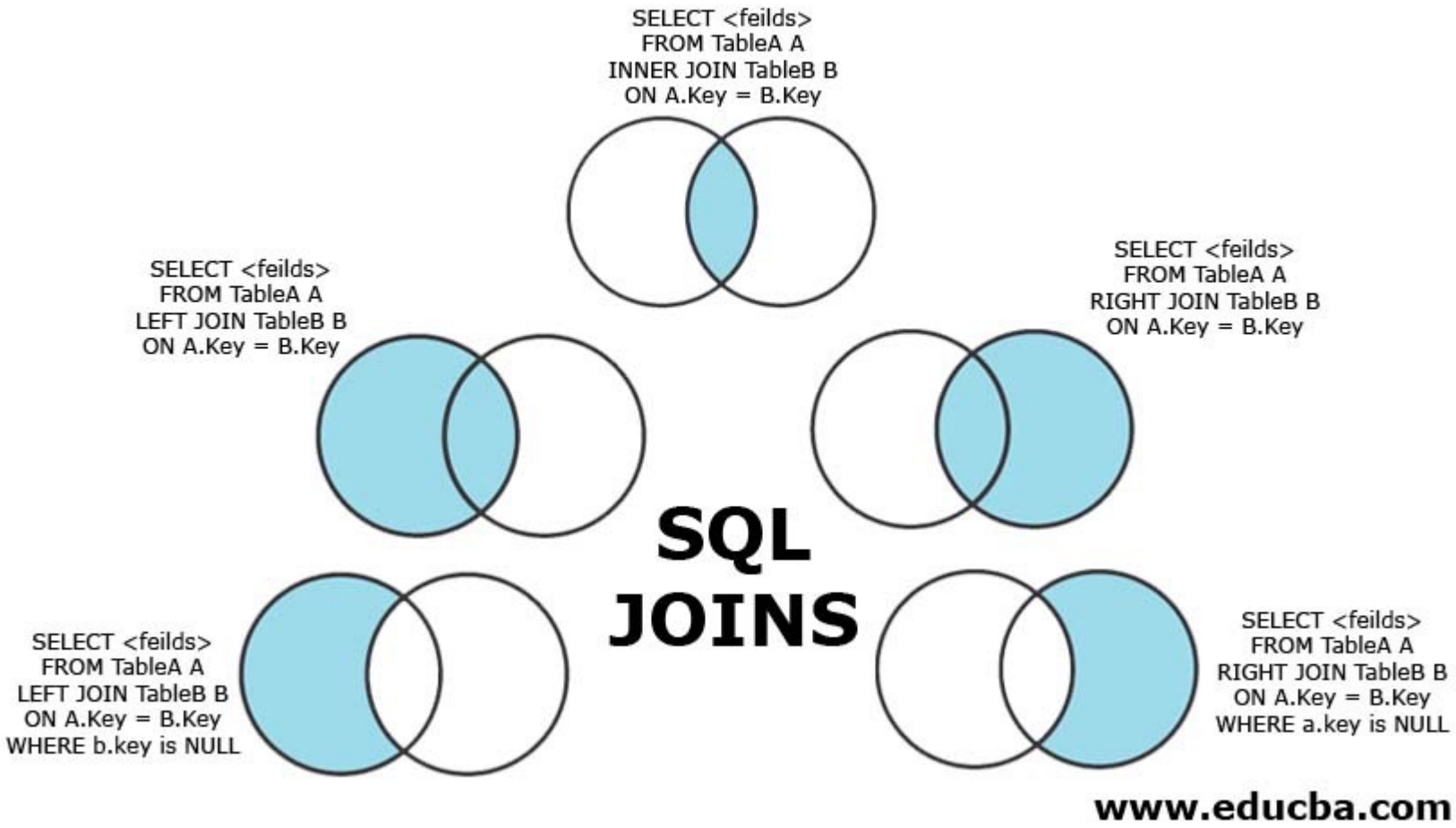

```
SELECT <select_list>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE A.Key IS NULL
```




```
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
```


SELECT <feilds>
FROM TableA A
INNER JOIN TableB B
ON A.Key = B.Key
SELECT <feilds>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
SELECT <feilds>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
SQL
JOINS
SELECT <feilds>
FROM TableA A
LEFT JOIN TableB B
ON A.Key = B.Key
WHERE b.key is NULL
SELECT <feilds>
FROM TableA A
RIGHT JOIN TableB B
ON A.Key = B.Key
WHERE a.key is NULL
www.educba.com

The following slides create tables for this enterprise data model

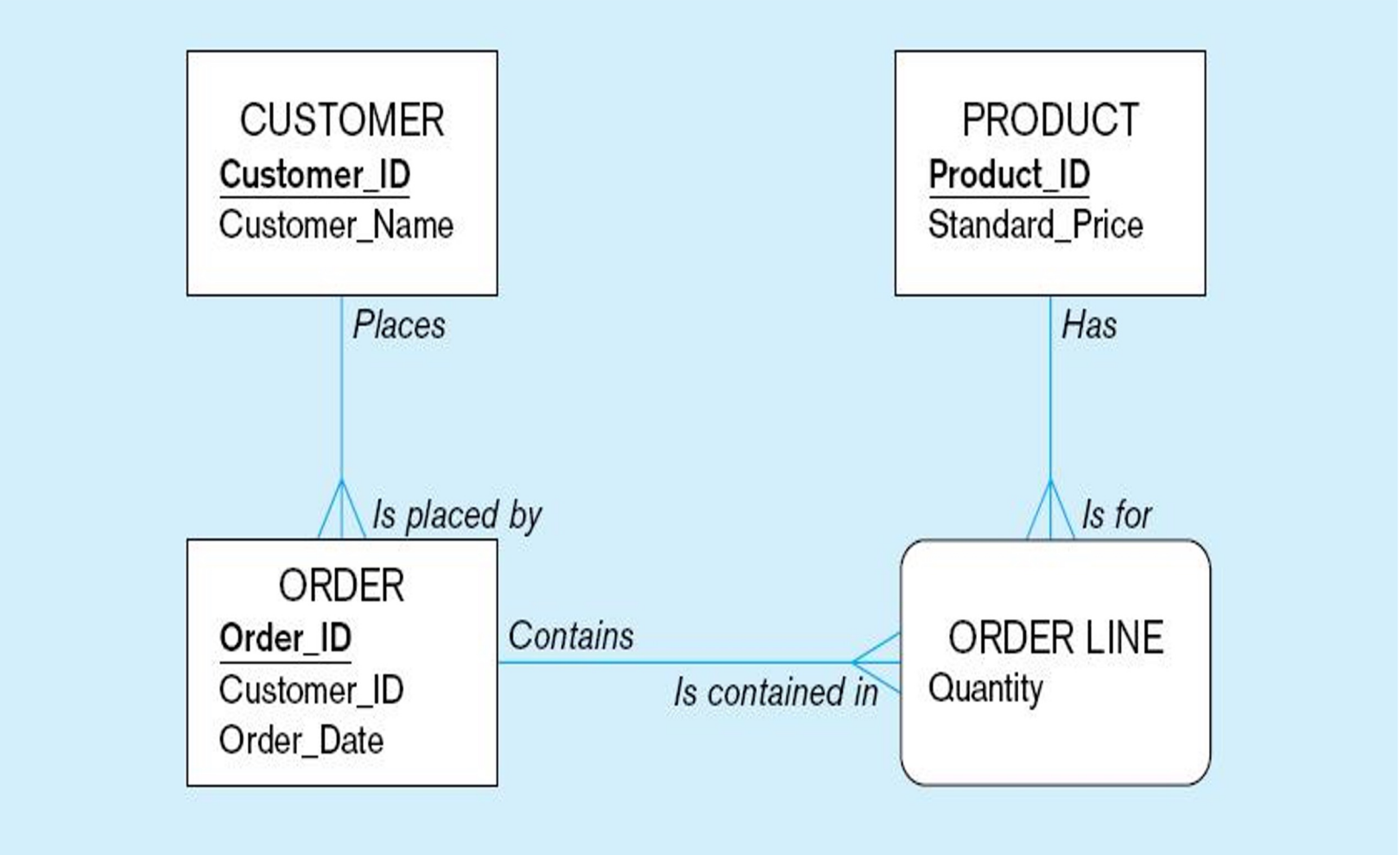
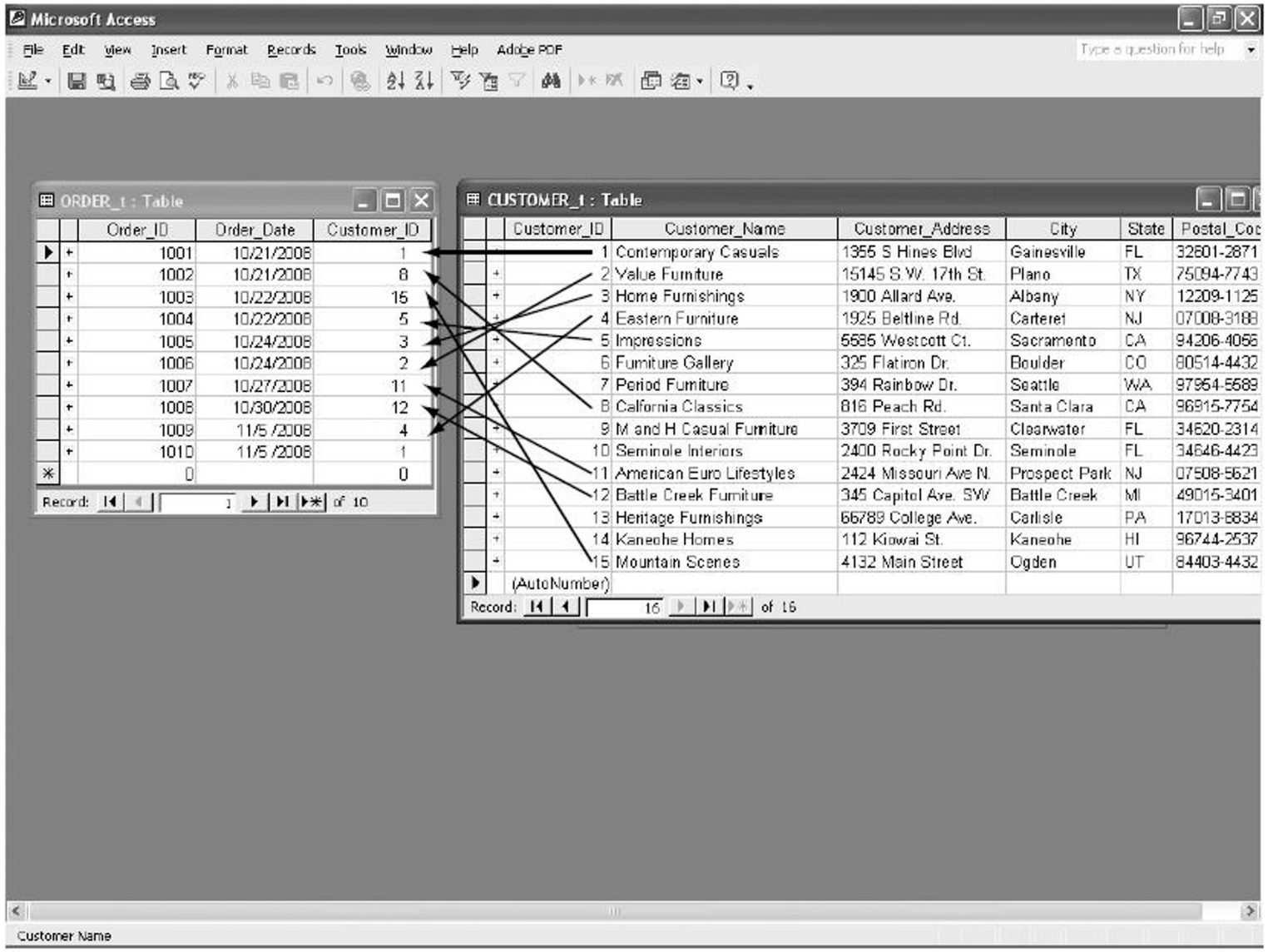


Figure 8-1 Pine Valley Furniture Company Customer and Order tables with pointers from customers to their orders



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.CUSTOMER_ID,    CUSTOMER_NAME, ORDER_ID
FROM CUSTOMER_T NATURAL JOIN ORDER_T ON
CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID;
```

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

Note: from Fig. 1, you see that only 10 Customers have links with orders

 Only 10 rows will be returned from this INNER join

Outer Join Example (Microsoft Syntax)

- 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.CUSTOMER_ID, CUSTOMER_NAME, ORDER_ID
FROM CUSTOMER_T, LEFT OUTER JOIN ORDER_T
ON CUSTOMER_T.CUSTOMER_ID = ORDER_T.CUSTOMER_ID;
```

LEFT OUTER JOIN syntax with ON causes customer data to appear even if there is no corresponding order data

Unlike INNER join, this will include customer rows with no matching order rows

Results

CUSTOMER_ID	CUSTOMER_NAME	ORDER_ID
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.

Unlike INNER join, this will include customer rows with no matching order rows

Multiple Table Join Example

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

Four tables involved in this join

```
SELECT CUSTOMER_T.CUSTOMER_ID, CUSTOMER_NAME,
   CUSTOMER_ADDRESS, CITY, SATE, POSTAL_CODE, ORDER_T.ORDER_ID,
   ORDER_DATE, QUANTITY, PRODUCT_DESCRIPTION, STANDARD_PRICE,
   (QUANTITY * UNIT_PRICE)
FROM CUSTOMER_T, ORDER_T, ORDER_LINE_T, PRODUCT_T

WHERE  CUSTOMER_T.CUSTOMER_ID = ORDER_LINE.CUSTOMER_ID  AND
   ORDER_T.ORDER_ID = ORDER_LINE_T.ORDER_ID
          AND ORDER_LINE_T.PRODUCT_ID = PRODUCT_PRODUCT_ID
          AND ORDER_T.ORDER_ID = 1006;
```

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

Figure 8-4 Results from a four-table join

From CUSTOMER_T table

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_ADDRESS	CUSTOMER_CITY	CUSTOMER_ST	POSTAL_CODE
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094 7743
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094 7743
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094 7743

ORDER_ID	ORDER_DATE	ORDERED_QUANTITY
1006	24-OCT-06	1
1006	24-OCT-06	2
1006	24-OCT-06	2

From ORDER_T table

PRODUCT_NAME	STANDARD_PRICE	(QUANTITY* STANDARD_PRICE)
Entertainment Center	650	650
Writer's Desk	325	650
Dining Table	800	1600

From PRODUCT_T table

Processing Multiple Tables Using Subqueries

- Subquery–placing an inner query (SELECT statement) inside an outer query
- Options:
  - In a condition of the WHERE clause
  - As a "table" of the FROM clause
  - Within the HAVING clause
- Subqueries can be:
  - Noncorrelated–executed once for the entire outer query
  - Correlated–executed once for each row returned by the outer query

Subquery Example

- Show all customers who have placed an order

The IN operator will test to see if the CUSTOMER_ID value of a row is included in the list returned from the subquery

```
SELECT CUSTOMER_NAME    FROM CUSTOMER_T
WHERE CUSTOMER_ID  IN
          (SELECT DISTINCT CUSTOMER_ID FROM ORDER_T);
```

Subquery is embedded in parentheses. In this case it returns a list that will be used in the WHERE clause of the outer query

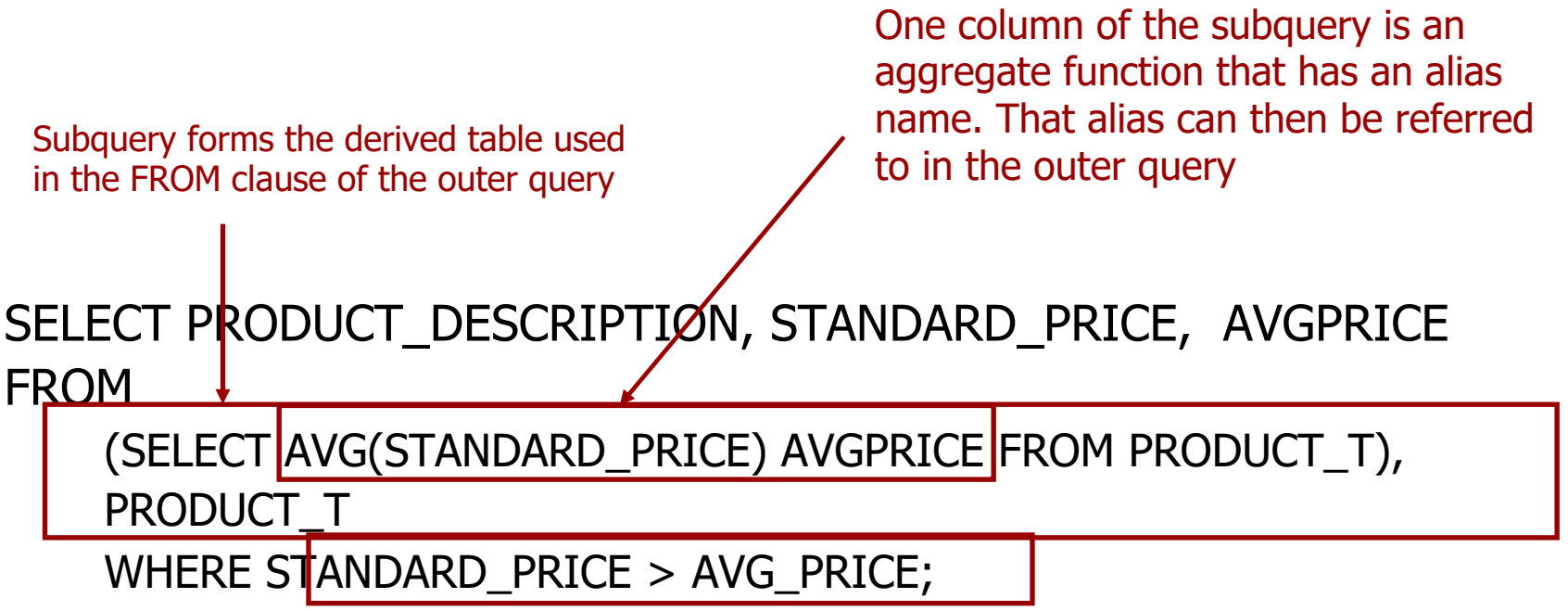
Result:

CUSTOMER_NAME
Contemporary Casuals
Value Furniture
Home Furnishings
Eastern Furniture
Impressions
California Classics
American Euro Lifestyles
Battle Creek Furniture
Mountain Scenes

9 rows selected.

Another Subquery Example

- Show all products whose standard price is higher than the average price



Union Queries

- Combine the output (union of multiple queries) together into a single result table

```
SELECT C1.CUSTOMER_ID,CUSTOMER_NAME,ORDERED_QUANTITY,
QUANTITY AS 'Largest Quantity'
  FROM CUSTOMER_T C1,ORDER_T O1, ORDER_LINE_T Q1
       WHERE C1.CUSTOMER_ID = O1.CUSTOMER_ID
       AND O1.ORDER_ID = Q1.ORDER_ID
       AND ORDERED_QUANTITY =
            (SELECT MAX(ORDERED_QUANTITY)
            FROM ORDER_LINE_T)
UNION
SELECT C1.CUSTOMER_ID,CUSTOMER_NAME,ORDERED_
QUANTITY, QUANTITY AS 'Smallest Quantity'
  FROM CUSTOMER_T C1,ORDER_T O1, ORDER_LINE_T Q1
       WHERE C1.CUSTOMER_ID = O1.CUSTOMER_ID
       AND O1.ORDER_ID = Q1.ORDER_ID
       AND ORDERED_QUANTITY =
            (SELECT MIN(ORDERED_QUANTITY)
            FROM ORDER_LINE_T)
ORDER BY ORDERED_QUANTITY;
```

Combine →

First query

Second query

Figure 8-7 Combining queries using UNION

```
SELECT C1.CUSTOMER_ID,CUSTOMER_NAME,ORDERED_QUANTITY, QUANTITY AS 'Largest Quantity'
FROM CUSTOMER_T C1,ORDER_T O1, ORDER_LINE_T Q1
WHERE C1.CUSTOMER_ID =O1.CUSTOMER_ID
AND O1.ORDER_ID =Q1.ORDER_ID
AND ORDERED_QUANTITY =
                (SELECT MAX(ORDERED_QUANTITY)
                FROM ORDER_LINE_T)
```

1. In the above query, the subquery is processed first and an intermediate results table created.
It contains the maximum quantity ordered from ORDER_LINE_T and has a value of 10.
2. Next the main query selects customer information for the customer or customers who ordered 10
of any item. Contemporary Casuals has ordered 10 of some unspecified item.

```
SELECT C1.CUSTOMER_ID,CUSTOMER_NAME,ORDERED_QUANTITY, QUANTITY AS 'Smallest Quantity'
FROM CUSTOMER_T C1,ORDER_T O1, ORDER_LINE_T Q1
WHERE C1.CUSTOMER_ID =O1.CUSTOMER_ID
AND O1.ORDER_ID =Q1.ORDER_ID
AND ORDERED_QUANTITY =
                (SELECT MIN(ORDERED_QUANTITY)
                FROM ORDER_LINE_T)
ORDER BY ORDERED_QUANTITY;
```

1. In the second main query, the same process is followed but the result returned is for the minimum order quantity.
2. The results of the two queries are joined together using the UNION command.
3. The results are then ordered according to the value in ORDERED_QUANTITY. The default is ascending value, so the orders with the smallest quantity, 1, are listed first.

Conditional Expressions Using Case Syntax

This is available with newer versions of SQL, previously not part of the standard

Figure 8-8

```
{CASE expression
{WHEN expression
THEN {expression  |  NULL}} . . .
| {WHEN predicate
THEN {expression  |  NULL}} . . .
[ELSE {expression  NULL}]
END }
| ( NULLIF (expression, expression) }
| ( COALESCE (expression . . .) }
```

Ensuring Transaction Integrity

- Transaction = A discrete unit of work that must be completely processed or not processed at all
  - May involve multiple updates
  - If any update fails, then all other updates must be cancelled
- SQL commands for transactions
  - BEGIN TRANSACTION/END TRANSACTION
    - Marks boundaries of a transaction
  - COMMIT
    - Makes all updates permanent
  - ROLLBACK
    - Cancels updates since the last COMMIT

Figure 8-9 An SQL Transaction sequence (in pseudocode)