Chapter 7
Advanced SQL

Objectives

- Define terms
- Write single and multiple table SQL queries
- Write noncorrelated and correlated subqueries
- Define and use five types of joins
  - (INNER) JOIN
  - EQUI-JOIN
  - NATURAL JOIN
  - OUTER JOIN
  - UNION JOIN

Nested Queries (Subqueries)

Subquery:

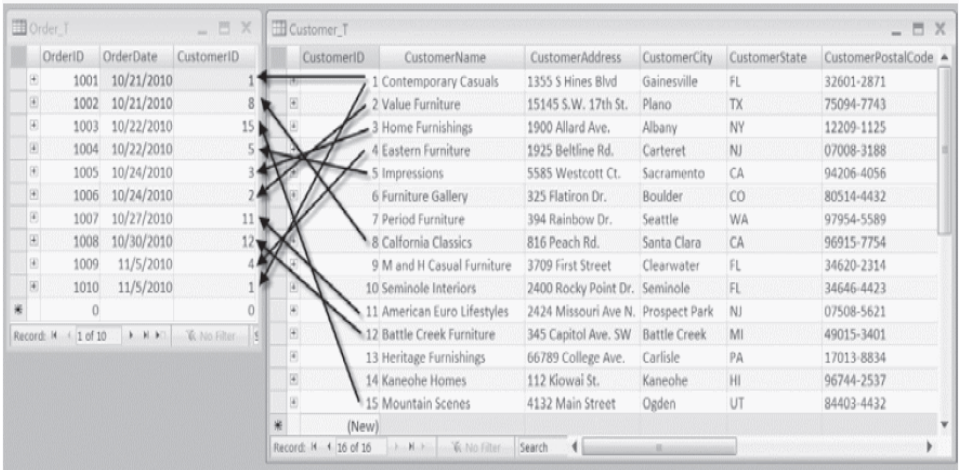
- Placing an inner query (SELECT statement) inside an outer query
- Can be used in place of a table name, or an expression that returns a list for operators that accept a list (IN, EXISTS, =ANY, etc.)

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 the names of all customers who placed an order



Subquery Example

Show all customers who have placed an order

```
SELECT CustomerName
  FROM Customer_T
    WHERE CustomerID IN
      (SELECT DISTINCT CustomerID
      FROM Order_T);
```

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

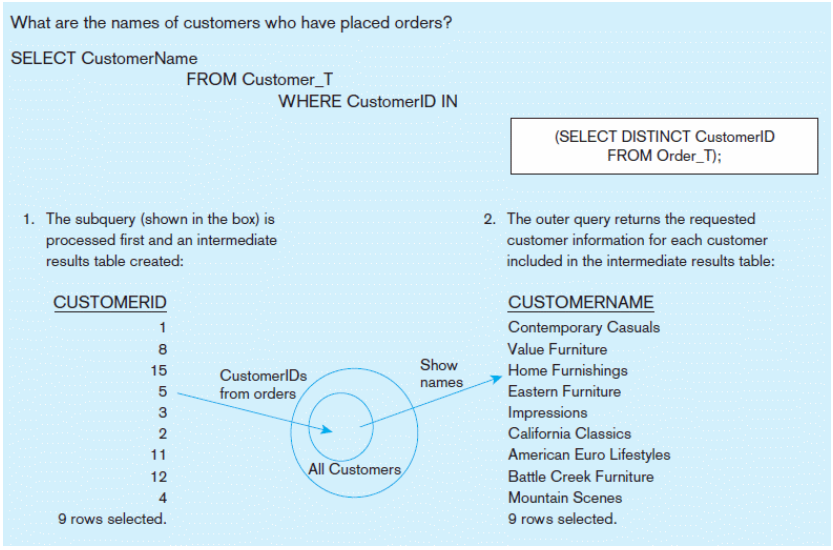
Subquery is embedded in parentheses. In this case it returns a list that will be used in the WHERE clause of the outer query

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.

Figure 7-7a Processing a noncorrelated subquery



A **NON-CORRELATED** subquery processes completely before the outer query begins

Practice: Exercise #5

List the names of all students not enrolled in any course during the first semester of 2008.

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
…	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

Set Membership: Existential

Show the names of all faculty who qualified to teach *some* course before 01/30/2013.

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
…	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
…		

Set Membership: Existential

Show all faculty who qualified to teach *some* course before 01/30/2013.

```
SELECT FacultyID, FacultyName
FROM    Faculty_T
WHERE FacultyID IN
( SELECT DISTINCT FacultyID
  FROM    Qualified_T
  WHERE DateQualified < '2013-01-30' );
```

Existential: Easy! ☺

Subquery returns the IDs of all faculty who qualified to teach some course before 01/30/2013

Set Membership: Universal

Show the names of all faculty who got *all* their qualifications before 01/30/2013.

Universal: Hard! ☹

Set Membership: Universal

Find *the other* faculty, who got *some* qualification *after* 01/30/2013.

```
SELECT FacultyName  FROM Faculty_T
WHERE  FacultyID   IN  ( SELECT DISTINCT FacultyID
                         FROM Qualified_T
                         WHERE DateQualified >= '2013-01-30' );
```

Find all faculty who got *all* qualifications *before* 10/30/2010.

```
SELECT FacultyName  FROM Faculty_T
WHERE  FacultyID   NOT IN  ( SELECT DISTINCT FacultyID
                             FROM Qualified_T
                             WHERE DateQualified >= '2013-01-30' );
```

Correlated vs. Noncorrelated Subqueries

Noncorrelated subqueries:

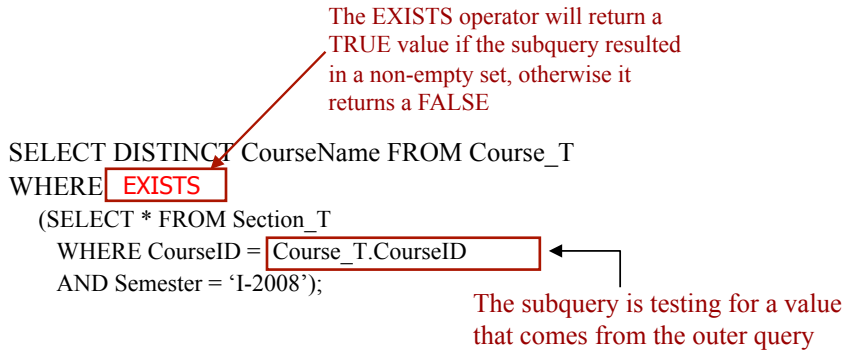
- Do not depend on data from the outer query
- Execute once for the entire outer query

Correlated subqueries:

- Make use of data from the outer query
- Execute once for each row of the outer query
- Can use the EXISTS operator

Correlated Subquery Example

Show all course names offered in the first semester of 2008.



➔ A correlated subquery always refers to an attribute from a table referenced in the outer query

Correlated Subquery Example

Show all courses offered in the first semester of 2008.

```
SELECT DISTINCT CourseName FROM Course_T
WHERE  EXISTS
   (SELECT * FROM Section_T
     WHERE CourseID = Course_T.CourseID
     AND Semester = 'I-2008');
```

1. The first CourseID is selected from Course_T (ISM 3113).
2. The subquery is evaluated to see if any sections exist for semester 'I-2008'. EXISTS is valued as TRUE and the CourseName is added to the results table.
3. The next CourseID is selected from Course_T (ISM 3112), etc.

Practice: Exercise #5

List the names of all students not enrolled in any course during the first semester of 2008. Give an alternative solution using a correlated query (with NOT EXISTS).

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
…	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

NULLs in SQL

- Whenever we don't have a value, we can use a NULL
- Can mean many things
  - Value does not exist
  - Value exists but is unknown
  - Value not applicable

- How does SQL cope with tables that have NULLs?

NULL Values

- If x is NULL then 4*(3-x)/7 is also NULL
- If x is NULL then x > 6 is UNKNOWN
- In SQL there are three boolean values:

FALSE = 0
UNKNOWN = 0.5
TRUE = 1

C1 AND C2 = min(C1, C2)
C1 OR C2 = max(C1, C2)
NOT C1 = 1 - C1

```
SELECT *
FROM Person
WHERE  (age < 25) AND
       (height > 6 OR weight > 190);
```

Rule in SQL: include only records that yields TRUE.

NULL Values

- Unexpected behavior:

```
SELECT *
FROM Person
WHERE  (age < 25) OR (age >= 25);
```

Some Persons not included!

- Can test for NULL explicitly:

```
SELECT *
FROM Person
WHERE  (age < 25) OR (age >= 25) OR (age IS NULL);
```

Quick Note: GROUP-BY vs. Subquery

- Find students who take >= 2 classes
- Attempt 1 (with subqueries):

```
SELECT StudentName
FROM Student_T
WHERE (SELECT COUNT(*) FROM Registration_T
        WHERE StudentId = Student_T.StudentID)  >= 2;
```

This is SQL by a novice

Quick Note (contd.)

- Find students who take >= 2 classes
- Attempt 2 (with GROUP BY):

```
SELECT      StudentName
FROM        Student_T AS S, Registration_T AS R
WHERE       S.StudentID = R.StudentID
GROUP BY    S.StudentID
HAVING      COUNT(R.SectionNo) >= 2;
```

This is SQL by an expert

Union Queries

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

```
SELECT C1.CustomerID, CustomerName, OrderedQuantity,
'Largest Quantity' AS Quantity
FROM Customer_T C1,Order_T O1, OrderLine_T Q1
      WHERE C1.CustomerID = O1.CustomerID
      AND O1.OrderID = Q1.OrderID
      AND OrderedQuantity =
      (SELECT MAX(OrderedQuantity)
      FROM OrderLine_T)
```

First query

Combine → **UNION**

```
SELECT C1.CustomerID, CustomerName, OrderedQuantity,
'Smallest Quantity'
FROM Customer_T C1, Order_T O1, OrderLine_T Q1
      WHERE C1.CustomerID = O1.CustomerID
      AND O1.OrderID = Q1.OrderID
      AND OrderedQuantity =
            (SELECT MIN(OrderedQuantity)
            FROM OrderLine_T)
ORDER BY 3;
```

Second query

Figure 7-8 Combining queries using UNION

```
SELECT C1.CustomerID, CustomerName, OrderedQuantity, 'Largest Quantity' AS Quantity
    FROM Customer_T C1,Order_T O1, OrderLine_T Q1
    WHERE C1.CustomerID = O1.CustomerID
            AND O1.OrderID = Q1.OrderID
            AND OrderedQuantity =
                        (SELECT MAX(OrderedQuantity)
                        FROM OrderLine_T)
```

1. In the above query, the subquery is processed first and an intermediate results table created. It contains the maximum quantity ordered from OrderLine_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.CustomerID, CustomerName, OrderedQuantity, 'Smallest Quantity'
    FROM Customer_T C1, Order_T O1, OrderLine_T Q1
    WHERE C1.CustomerID = O1.CustomerID
            AND O1.OrderID = Q1.OrderID
            AND OrderedQuantity =
                        (SELECT MIN(OrderedQuantity)
                        FROM OrderLine_T)
ORDER BY 3;
```

Note: with UNION queries, the number and data types of the attributes in the SELECT clauses of both queries must be identical

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 OrderedQuantity. The default is ascending value, so the orders with the smallest quantity, 1, are listed first.

Processing Multiple Tables–JOINs

JOIN – Cartesian product of two tables (which gets us all pairs of rows, with one coming from each table) followed by a selection process according to some predicate.

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

STUDENT S JOIN REGISTRATION R

S.StudentID	S.StudentName	**R.StudentID**	**R.SectionNo**	**R.Semester**
38214	Letersky	38214	2714	I-2008
38214	Letersky	54907	2714	I-2008
38214	Letersky	54907	2715	I-2008
38214	Letersky	66324	2713	I-2008
54907	Altvater	38214	2714	I-2008
54907	Altvater	54907	2714	I-2008
54907	Altvater	54907	2715	I-2008
54907	Altvater	66324	2713	I-2008
66324	Aiken	38214	2714	I-2008
66324	Aiken	54907	2714	I-2008
66324	Aiken	54907	2715	I-2008
66324	Aiken	66324	2713	I-2008
70542	Marra	38214	2714	I-2008
70542	Marra	54907	2714	I-2008
70542	Marra	54907	2715	I-2008
70542	Marra	66324	2713	I-2008

EQUI-JOIN

EQUI-JOIN – two or more tables appear in the FROM clause, separated by commas; the join condition is added to the WHERE clause.

Example: For each student registered in the first semester of 2008, what is the student name and ID?

```
SELECT DISTINCT   S.StudentID, StudentName
FROM              Student_T AS S, Registration_T AS R
WHERE             S.StudentID = R.StudentID;
```

JOIN between Student_T and Registration_T

Practice: Exercise #1

a. Display the course ID and course name for all courses with an ISM prefix.

b. Display all courses for which Professor Berndt has been qualified.

c. Display the class roster, including student name, for all students enrolled in section 2714 of ISM 4212.

Practice: Exercise #2

List the names of all instructors qualified to teach ISM 3113.

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
...		

Practice: Exercise #3

Is any instructor qualified to teach ISM 3113 and not qualified to teach ISM 4930? List their names.

FACULTY (**FacultyID**, FacultyName)

FacultyID	FacultyName
2143	Birkin
3467	Berndt
4756	Collins
...	

QUALIFIED (**FacultyID**, **CourseID**, DateQualified)

FacultyID	**CourseID**	DateQualified
2143	ISM 3112	9/1988
2143	ISM 3113	9/1988
3467	ISM 4212	9/1995
3467	ISM 4930	9/1996
4756	ISM 3113	9/1991
4756	ISM 3112	9/1991
...		

Practice: Exercise #4

a. How many students are enrolled in section 2714 during semester I-2008?

b. How many students are enrolled in ISM 3113 during semester I-2008?

STUDENT (**StudentID**, StudentName)

StudentID	StudentName
38214	Letersky
54907	Altvater
66324	Aiken
70542	Marra
...	

REGISTRATION (**StudentID**, **SectionNo**, **Semester**)

StudentID	**SectionNo**	**Semester**
38214	2714	I-2008
54907	2714	I-2008
54907	2715	I-2008
66324	2713	I-2008

EQUI-JOIN – Alternative Syntax

For each student registered in the first semester of 2008, what is the student name and ID?

```
SELECT DISTINCT   S.StudentID, StudentName
FROM              Student_T  S JOIN  Registration_T R  ON
                  (S.StudentID = R.StudentID);
```

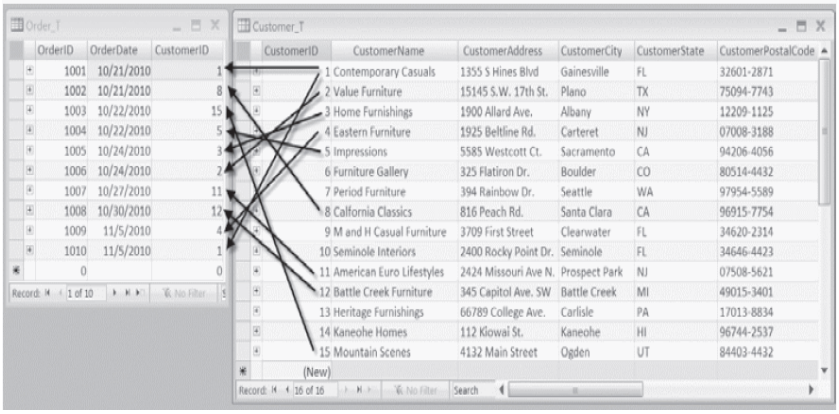
Selection Condition

(INNER) JOIN clause is an alternative to WHERE clause, and is used to match primary and foreign keys.

An INNER join will only return rows from each table that have matching rows in the other.

Implicit or Explicit EQUI-JOIN?

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



Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2010	1
1002	10/21/2010	8
1003	10/22/2010	15
1004	10/22/2010	5
1005	10/24/2010	3
1006	10/24/2010	2
1007	10/27/2010	11
1008	10/30/2010	12
1009	11/5/2010	4
1010	11/5/2010	1
0		0

Record: 1 of 10

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: 16 of 16 Search

Implicit EQUI-JOIN:

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

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

Result:

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

Explicit EQUI-JOIN:

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

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

Same Result:

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

Natural JOIN

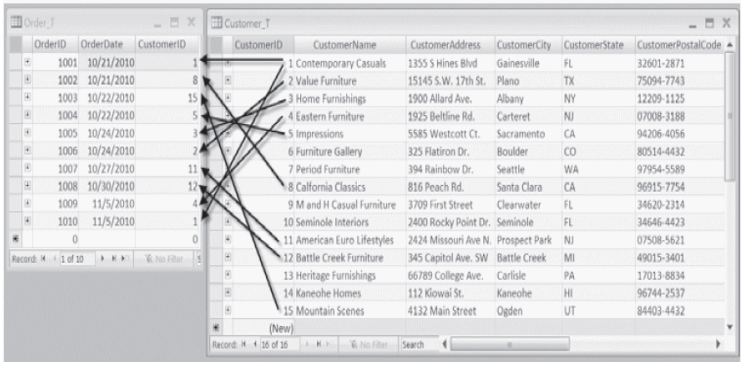
Most JOIN conditions involve the equality predicate. SQL supports the concept of a NATURAL JOIN, which is a join in which the join condition is created implicitly, by requiring equality of all fields with the same name in both tables.

```
SELECT DISTINCT    StudentID,  StudentName
FROM               Student_T  NATURAL JOIN Registration_T;
```

StudentID is a field common to both tables.
Only registered students will be listed.

Natural JOIN:

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



Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2010	1
1002	10/21/2010	8
1003	10/22/2010	15
1004	10/22/2010	5
1005	10/24/2010	3
1006	10/24/2010	2
1007	10/27/2010	11
1008	10/30/2010	12
1009	11/5/2010	4
1010	11/5/2010	1
0		0

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	Calfornia 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)				

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

Natural JOIN and USING

Natural joins are too brittle. SQL also supports a much better JOIN ... USING syntax. Rather than writing ON and the join condition, we write USING and then a list of fields which must match.

```
SELECT      StudentID,  StudentName
FROM        Student_T  JOIN Registration_T USING (StudentID);
```

Natural JOIN Flaw

Natural joins in SQL are semantically equivalent to ***equi-joins***, meaning that rows must have matching values to appear in the result table.

Suppose we want a list of all students, and their registration information, if any.

```
SELECT      StudentID,  StudentName
FROM        Student_T  JOIN Registration_T USING (StudentID);

/* Students who never registered will be lost */
```

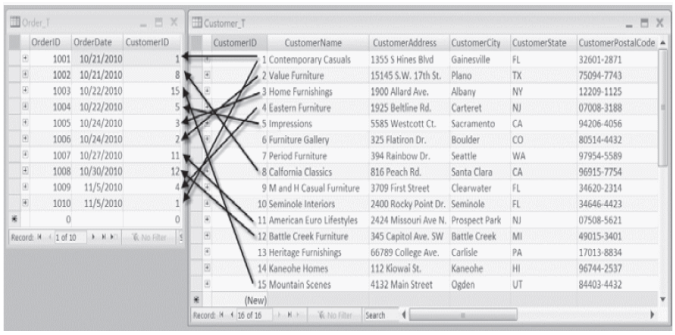
We want *all* rows in the result, even when there is no matching field on the other table. This can be achieved with OUTER JOIN.

Outer JOIN

- JOIN in which rows with no matching values in common columns are included in the result table
  - LEFT OUTER JOIN: Include the left row even if there is no match in the right table
  - RIGHT OUTER JOIN: Include the right row even if there is no match in the left table
  - FULL OUTER JOIN: Include both left and right row even if there is no match

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
ON        Customer_T.CustomerID  = Order_T.CustomerID;
```

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.

Outer JOIN Practice

List the student ID, name and registration information for each student. Include students even if not yet registered.

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.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;
```

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)?

Employee_T

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	

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:

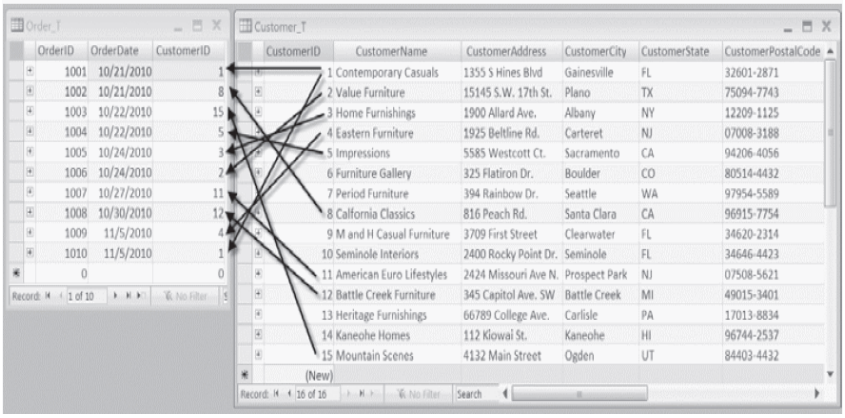
EMPLOYEEID	EMPLOYEENAME	MANAGER
123-44-347	Jim Jason	Robert Lewis

Self-joins are usually used on tables with unary relationships

JOIN vs. Subquery

Some queries could be accomplished by either a join or a subquery.

Query: What are the name and address of the customer who placed order number 1008?



Order_T

OrderID	OrderDate	CustomerID
1001	10/21/2010	1
1002	10/21/2010	8
1003	10/22/2010	15
1004	10/22/2010	5
1005	10/24/2010	3
1006	10/24/2010	2
1007	10/27/2010	11
1008	10/30/2010	12
1009	11/5/2010	4
1010	11/5/2010	1
0		0

Record: 1 of 10

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	Calfornia 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 Kowai St.	Kaneohe	HI	96744-2537
15	Mountain Scenes	4132 Main Street	Ogden	UT	84403-4432
(New)					

Record: 16 of 16 Search

JOIN vs. Subquery

Some queries could be accomplished by either a join or a subquery.

Query: What are the name and address of the customer who placed order number 1008?

Join version

```
SELECT CustomerName, CustomerAddress, CustomerCity,
   CustomerState, CustomerPostalCode
FROM Customer_T, Order_T
WHERE Customer_T.CustomerID = Order_T. CustomerID
   AND OrderID = 1008;
```

Subquery version

```
SELECT CustomerName, CustomerAddress, CustomerCity,
CustomerState, CustomerPostalCode
   FROM Customer_T
      WHERE Customer_T.CustomerID =
        (SELECT Order_T.CustomerID
           FROM Order_T
              WHERE OrderID = 1008);
```