Database Systems

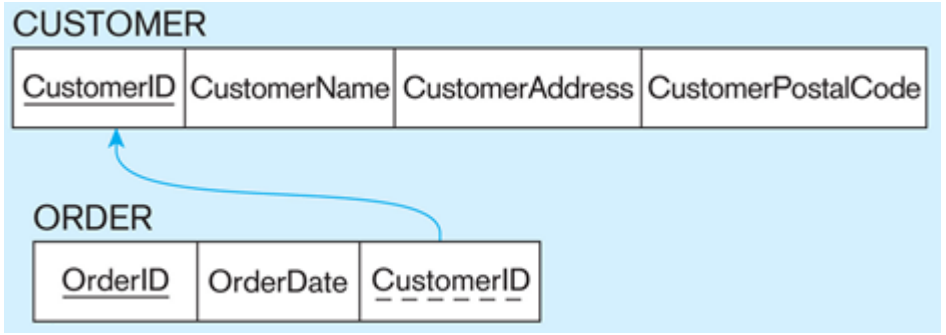
COMP 3010E FALL 2025

LECTURE 7 ADVANCED SQL

Advanced SQL

❑ Processing Multiple Tables

- ❖ JOIN
  - ○ Cross join
  - ○ Equi-join
  - ○ Natural Join
  - ○ Outer Join*
  - ○ Self-join*
- ❖ Subqueries
- ❖ Set-based Operations: UNION, INTERSECT, MINUS



*** Will be discussed later. Not included in HW or exam.**

Relationships and Joins

- Relationships between tables in RDBMS are created using common fields (FK) in related tables
- We can use values in the common fields (or fields with common domains) to integrate information from multiple tables
- The operation of bringing the data from two tables into a single result table is called a **join**

Join

A relational operation that combines two tables with a common domain into a single table

Student

Sid	**Name**	**Age**	**dno**
1	James	25	1
2	Lucas	24	1
3	Alice	21	2

Department

dno	**Name**	**Head**
1	Management Sciences	Nick
2	Accounting	Kate

dno in Student is a foreign key

Equi-Join

A join with the joining condition based on equality between values in the common columns. Common columns appear (redundantly) in the result table.

```
SELECT *
FROM Student, Department
WHERE Student.dno = Department.dno;
```

All necessary tables

Use <Table>.<Column> to avoid confusion

Join conditions

Sid	Name	Age	dno	dno	Name	Head
1	James	25	1	1	Management Sciences	Nick
2	Lucas	24	1	1	Management Sciences	Nick
3	Alice	21	2	2	Accounting	Kate

Another Example – Company Data

- Find out information of the customer who placed order # 1008

  Customer_T (**CustomerID**, CustomerName, CustomerAddress, …, …)
  Order_T(**OrderID**, CustomerID, OrderDate)

What if you are asked to display only OrderID, CustomerID, and CustomerName?

Rename a table in the query

```
SELECT *
FROM Order_T o, Customer_T c
WHERE o.Customerid = c.Customerid
AND   o.Orderid = 1008
```

Can we SELECT OrderID, CustomerID, CustomerName?

SELECT OrderID, c.CustomerID, CustomerName
FROM ….

ORDERID	CUSTOMERID	ORDERDATE	ORDERDUE	CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTALCODE
1001	1	10/21/2010	-	1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
1010	1	11/05/2010	-	1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
1006	2	10/24/2010	-	2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
1005	3	10/24/2010	-	3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
1009	4	11/05/2010	-	4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188
1004	5	10/02/2010	-	5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056
1002	8	10/21/2010	-	8	California Classics	816 Peach Rd.	Santa Clara	CA	96915-7754
1007	11	10/27/2010	-	11	American Euro Lifestyles	2424 Missouri Ave N.	Prospect Park	NJ	07508-5621
1008	12	10/30/2010	-	12	Battle Creek Furniture	345 Capitol Ave. SW	Battle Creek	MI	49015-3401
1003	15	10/02/2010	-	15	Mountain Scenes	4132 Main Street	Ogden	UT	84403-4432

A different way to do JOIN

- Establish an equi-join in the FROM clause

```
SELECT *
FROM Student JOIN Department ON
     Student.dno = Department.dno;
```

=

```
SELECT *
FROM Student, Department
WHERE Student.dno = Department.dno;
```

Natural Join

Same as an equi-join except that one of the duplicate columns is eliminated in the result table

```
SELECT *
FROM Student NATURAL JOIN Department ON
     Student.dno = Department.dno;
```

No duplicate

Sid	Name	Age	dno	Name	Head
1	James	25	1	Management Sciences	Nick
2	Lucas	24	1	Management Sciences	Nick
3	Alice	21	2	Accounting	Kate

Cross Join

- Also called **Cartesian** join
  - No join condition specified (does not reflect the relationship)
  - Generate all combinations of rows in two tables – a Cartesian product
  - Create a large table without useful or meaningful results

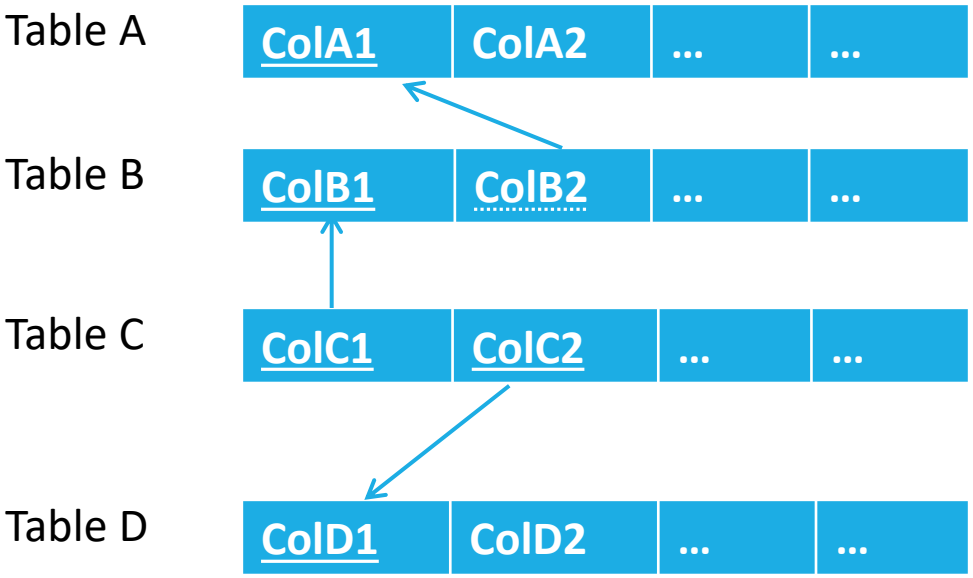
```
SELECT *
FROM Student, Department;
```

Do all results make sense?

Sid	Name	Age	dno	dno	Name	Head
1	James	25	1	1	Management Sciences	Nick
2	Lucas	24	1	1	Management Sciences	Nick
3	Alice	21	2	1	Management Sciences	Nick
1	James	25	1	2	Accounting	Kate
2	Lucas	24	1	2	Accounting	Kate
3	Alice	21	2	2	Accounting	Kate

Join Multiple Tables (1)

- To join more than two tables, simply add more tables to the FROM clause and add additional conditions (with AND) to the WHERE clause




```
SELECT *
FROM A, B, C, D
WHERE A.ColA1 = B.ColB2
AND   B.ColB1 = C.ColC1
AND   C.ColC2 = D.ColD1
```

Each pair of related tables should be joined. the number of joining equality conditions is: [number of tables – 1] in most cases.

Join Multiple Tables (2)

Query: Display the customer info associated the product IDs that the customer purchased

- Customer_T:

CustomerID	CustomerName	… …

- Order_T:

OrderID	CustomerID	OrderDate

- OrderLine_T:

OrderID	ProductID	OrderQuantity

The JOIN part

```
SELECT CustomerName, ProductID
FROM   Customer_T, Order_T, OrderLine_T
WHERE Customer_T.CustomerID = Order_T.CustomerID
AND      Order_T.OrderID = OrderLine_T.OrderID;
```

CUSTOMERNAME	PRODUCTID
Contemporary Casuals	1
Contemporary Casuals	2
Contemporary Casuals	4
Contemporary Casuals	8
Value Furniture	4
Value Furniture	5
Value Furniture	7
Home Furnishings	4
Eastern Furniture	4
Eastern Furniture	7
Impressions	6
Impressions	8
California Classics	3
American Euro Lifestyles	1
American Euro Lifestyles	2
Battle Creek Furniture	3
Battle Creek Furniture	8
Mountain Scenes	3

JOIN Operation

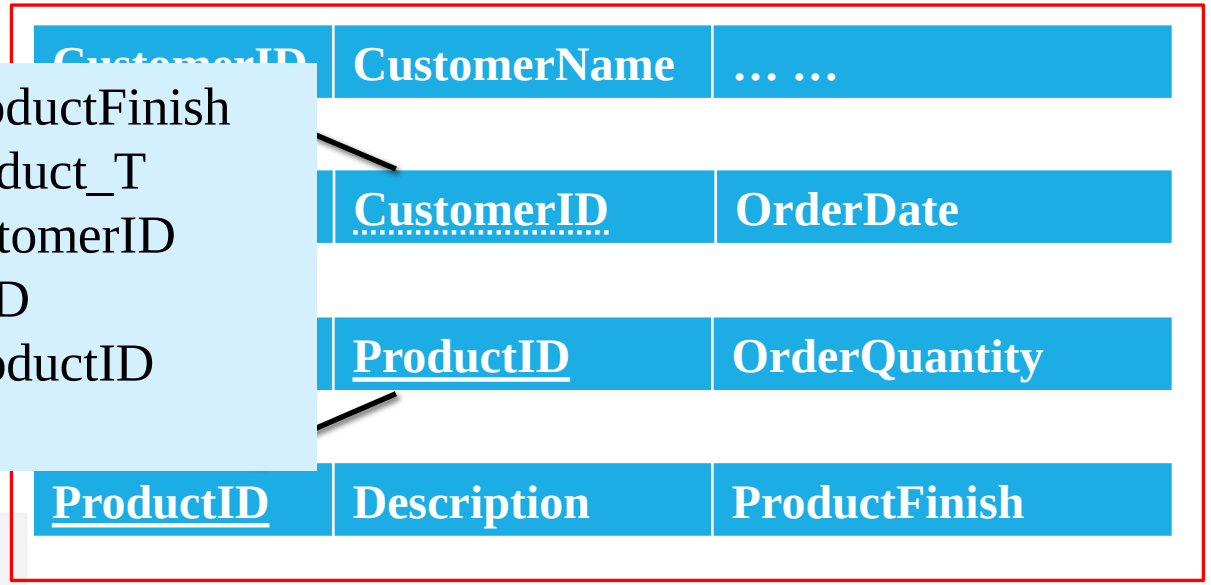
```
SELECT CustomerName, ProductID
FROM   Customer_T, Order_T, OrderLine_T
WHERE Customer_T.CustomerID = Order_T.CustomerID
AND      Order_T.OrderID = OrderLine_T.OrderID;
```

	CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTALCODE	ORDERID	CUSTOMERID_1	ORDERDATE	ORDERID_1	PRODUCTID	ORDEREDQUANTITY
1	1	Contemporar...	1355 S Hines Blvd	Gainesville	FL	32601-2871	1001	1	10-21-2010	1001	1	2
2	1	Contemporar...	1355 S Hines Blvd	Gainesville	FL	32601-2871	1001	1	10-21-2010	1001	2	2
3	1	Contemporar...	1355 S Hines Blvd	Gainesville	FL	32601-2871	1001	1	10-21-2010	1001	4	1
4	8	California ...	816 Peach Rd.	Santa Clara	CA	96915-7754	1002	8	10-22-2013	1002	3	5
5	15	Mountain Sc...	4132 Main Street	Ogden	UT	84403-4432	1003	15	10-23-2010	1003	3	3
6	5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056	1004	5	10-02-2014	1004	6	2
7	5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056	1004	5	10-02-2014	1004	8	2
8	3	Home Furnis...	1900 Allard Ave.	Albany	NY	12209-1125	1005	3	10-24-2015	1005	4	3
9	2	Value Furni...	15145 S.W. 17t...	Plano	TX	75094-7743	1006	2	10-24-2010	1006	4	1
10	2	Value Furni...	15145 S.W. 17t...	Plano	TX	75094-7743	1006	2	10-24-2010	1006	5	2

Join Multiple Tables (3)

Query: Find the names of all of the customers along with the description and finish of the products that they have purchased.

```
SELECT   CustomerName, ProductDescription, ProductFinish
FROM     Customer_T, Order_T, OrderLine_T, Product_T
WHERE    Customer_T.CustomerID = Order_T.CustomerID
AND      Order_T.OrderID = OrderLine_T.OrderID
AND      OrderLine_T.ProductID = Product_T.ProductID
ORDER BY CustomerName;
```



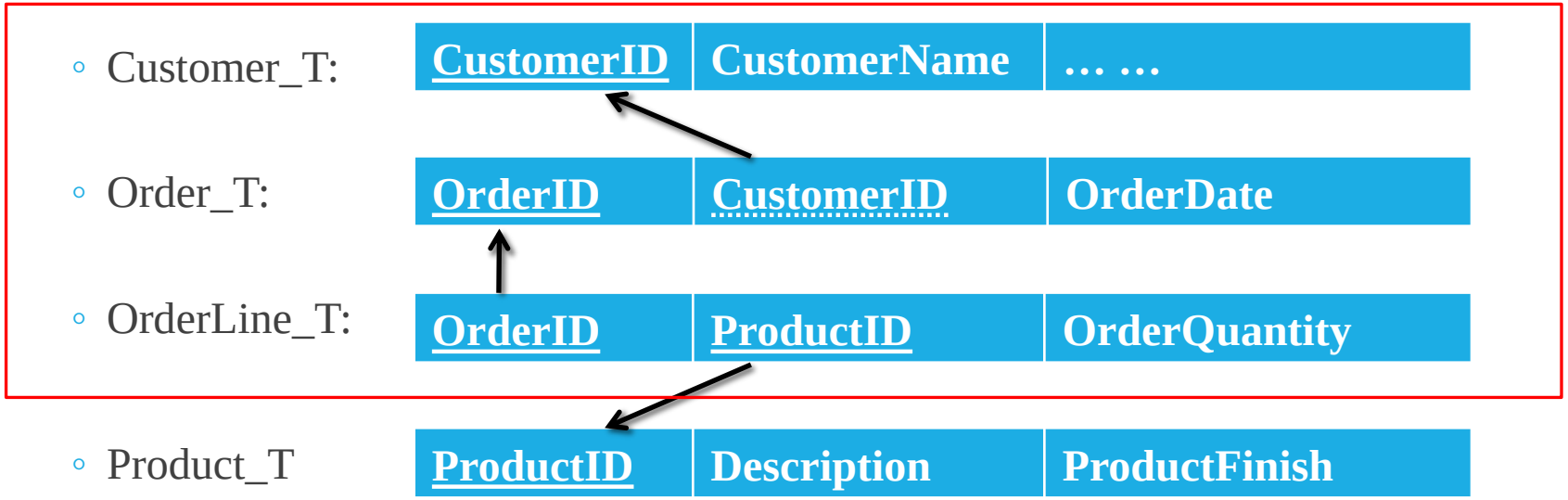
CUSTOMERNAME	PRODUCTDESCRIPTION	PRODUCTFINISH
American Euro Lifestyles	End Table	Cherry
American Euro Lifestyles	Coffee Table	Natural Ash
Battle Creek Furniture	Computer Desk	Walnut
Battle Creek Furniture	Computer Desk	Natural Ash
California Classics	Computer Desk	Natural Ash

partial data…

Join Multiple Tables (4)

Query: Find the names of all of the customers that have purchased more than one product.
Also, provide the number of unique products that such customers have purchased.

Which tables do we need to answer the query?



Join Multiple Tables (5)

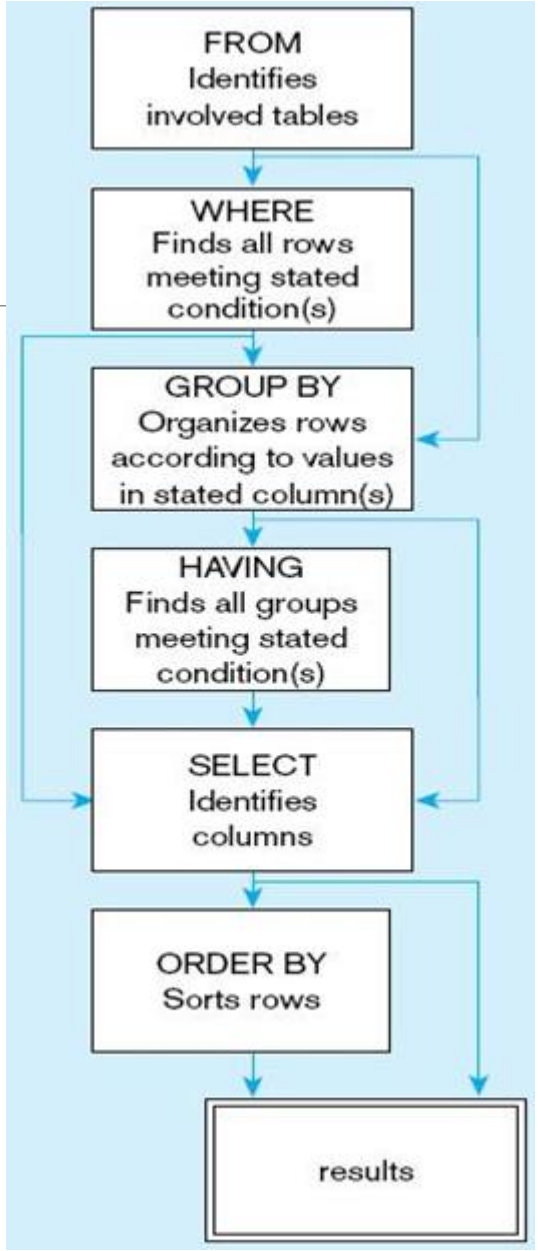
```
SELECT      CustomerName, COUNT(DISTINCT ProductID) AS Number_Of_Products
FROM        Customer_T, Order_T, OrderLine_T
WHERE       Customer_T.CustomerID = Order_T.CustomerID
AND         Order_T.OrderID = OrderLine_T.OrderID
GROUP BY    CustomerName
HAVING      COUNT(DISTINCT ProductID) > 1
ORDER BY    Number_Of_Products DESC, CustomerName;
```

	CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTA
1	1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
2	1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
3	1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
4	1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871
5	2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
6	2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
7	2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743
8	3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125
9	4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188

CUSTOMERNAME	NUMBER_OF_PRODUCTS
Contemporary Casuals	4
Value Furniture	3
American Euro Lifestyles	2
Battle Creek Furniture	2
Eastern Furniture	2
Impressions	2

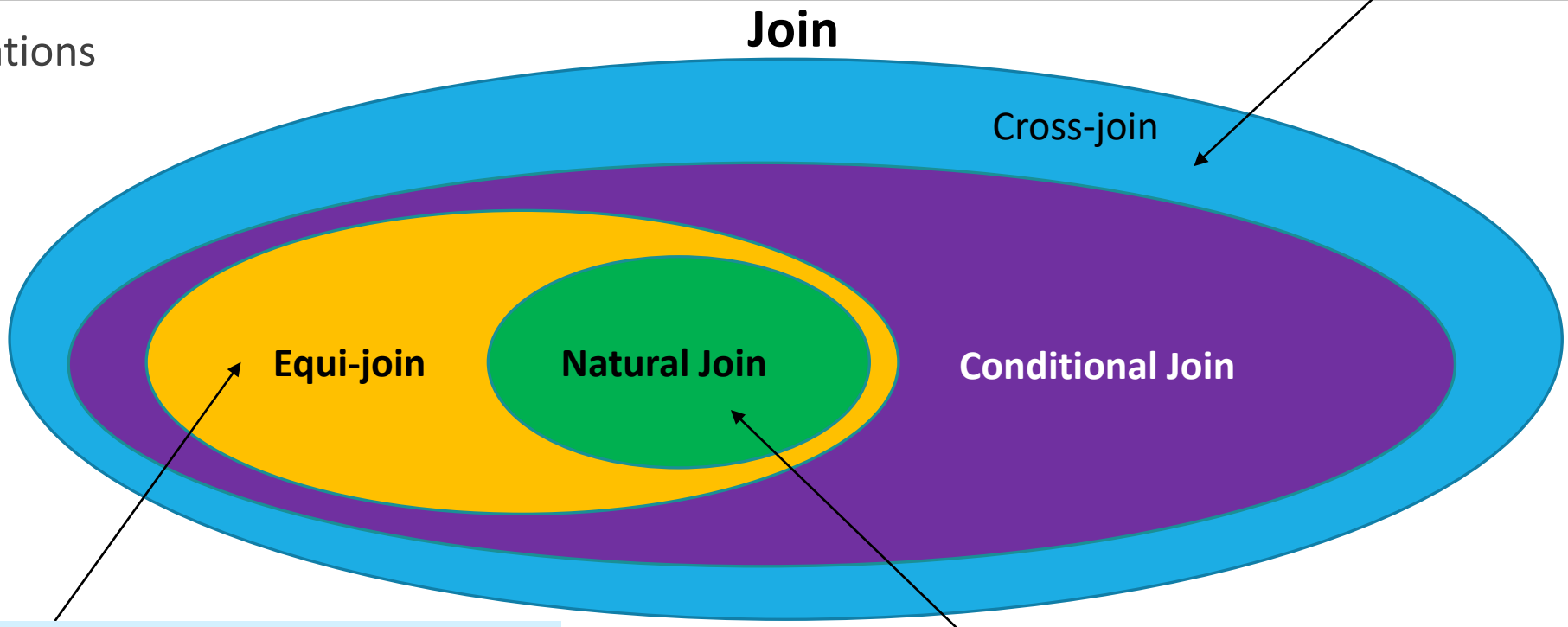
Agenda

- HW2 posted (Due Tue 10/28, midnight)
- Lab 2 report, Oracle Database Installation
- Review SQL 6 clauses and Join



Quick Review

JOIN operations




```
SELECT *
FROM Student, Department
```

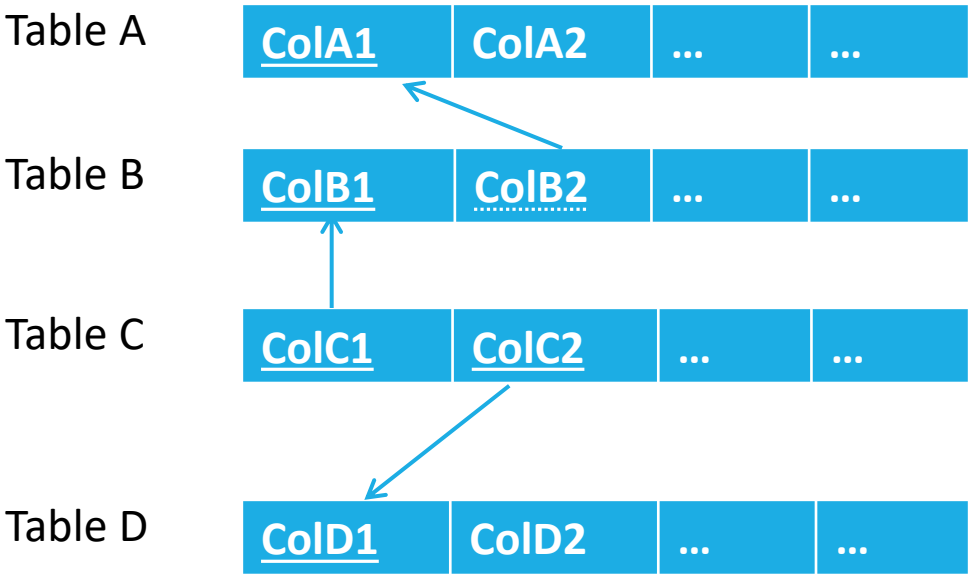
```
SELECT *
FROM Student, Department
WHERE Student.dno = Department.dno;
```

```
SELECT *
FROM Student NATURAL JOIN Department ON dno;

-- Join column must exist in both tables w/ same data type
```

Join Multiple Tables

- To join more than two tables, simply add more tables to the FROM clause and add additional conditions (with AND) to the WHERE clause




```
SELECT *
FROM A, B, C, D
WHERE A.ColA1 = B.ColB2
AND   B.ColB1 = C.ColC1
AND   C.ColC2 = D.ColD1
```

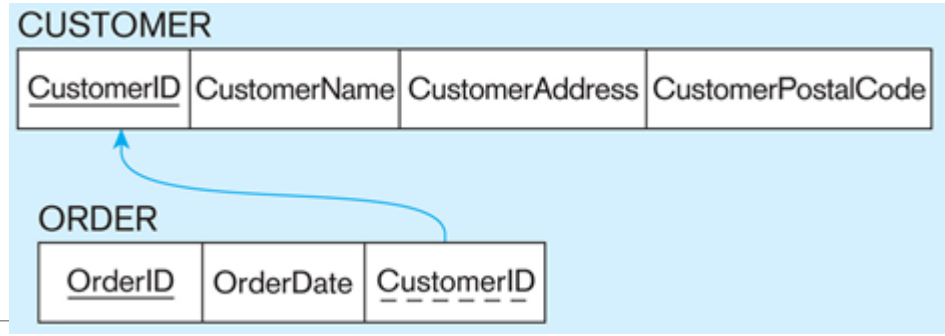
Each pair of related tables should be joined. the number of joining equality conditions is: [number of tables – 1] in most cases.

Exercise

1. For all students that take classes in the building of ‘T5’, find the number of students in each department.


```
SELECT   Dept, COUNT(DISTINCT S.StudentID)
FROM     Student S, Registration R, Course C
WHERE    S.StudentID = R.StudentID
AND      R.CourseID = Course.CourseID
AND      Building = ‘T5’
GROUP BY Dept;
```

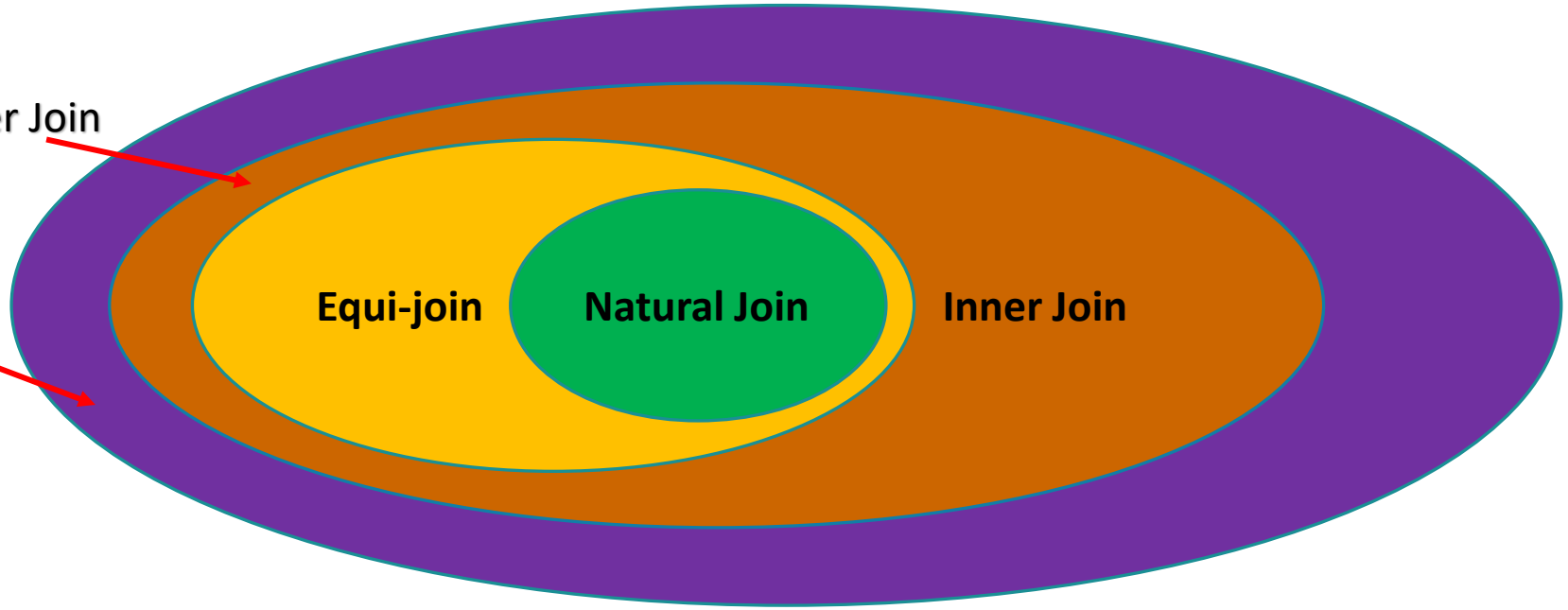
JOIN



- JOIN
  - Cross join
  - Equi-join
  - Natural Join

  Inner Join
  - Outer Join

Inner join: keep only matched rows from both sides



Inner Join

If you use cross-join, how many rows will you get in the result table?

Sid	Name	Age	dno
1	James	25	1
2	Lucas	24	1
3	Alice	21	2
4	Bruce	30	

dno	Name	Head
1	Management Sciences	Nick
2	Accounting	Kate
3	Computer Science	Alberto

Student without a matching department

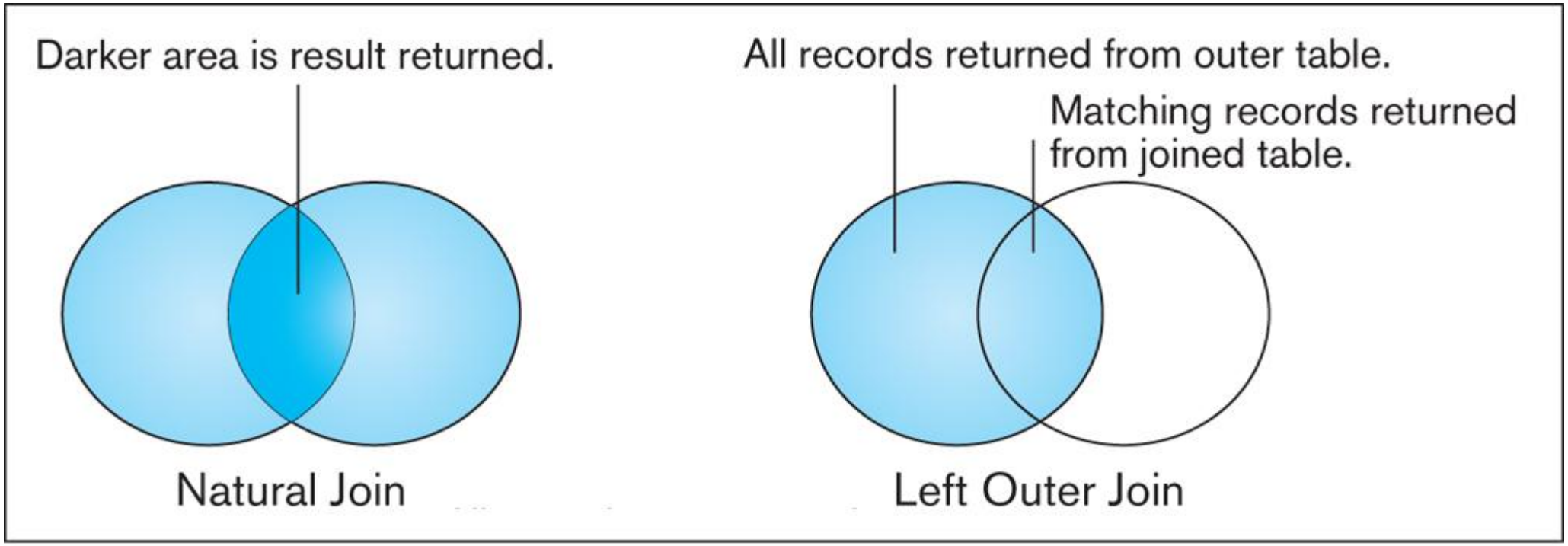
Department without a matching student

Only MATCHED rows are selected

Sid	Name	Age	dno	dno	Name	Head
1	James	25	1	1	Management Sciences	Nick
2	Lucas	24	1	1	Management Sciences	Nick
3	Alice	21	2	2	Accounting	Kate

Outer Join

Rows that do not have matching values in common columns are also included in the result table. Null values appear in columns where there is not a match.



Outer Join

Rows that do not have matching values in common columns are also included in the result table. Null values appear in columns where there is not a match.

Student

Sid	**Name**	**Age**	**dno**
1	James	25	1
2	Lucas	24	1
3	Alice	21	2
4	Bruce	30	

Department

dno	**Name**	**Head**
1	Management Sciences	Nick
2	Accounting	Kate

Left Outer Join (1)

NOTE: the left-right order of the two joining tables does NOT matter for cross-join, equi-join, inner-join, or natural-join.

Return all rows of the left-hand-side table no matter whether there is a match or not

Display **student** info associated with their **department** info. For those students who have not been assigned to any department, leave the department-related columns with NULL values

```
SELECT *
FROM Student LEFT OUTER JOIN Department ON
     Student.dno = Department.dno;
```

The keyword OUTER can be omitted

Left Outer Join (2)

Sid	Name	Age	dno
1	James	25	1
2	Lucas	24	1
3	Alice	21	2
4	Bruce	30	

dno	Name	Head
1	Management Sciences	Nick
2	Accounting	Kate
3	Computer Science	Alberto

Sid	Name	Age	dno	dno	Name	Head
1	James	25	1	1	Management Sciences	Nick
2	Lucas	24	1	1	Management Sciences	Nick
3	Alice	21	2	2	Accounting	Kate
4	Bruce	30				

Leave department-related columns with NULL values

Right Outer Join (1)

Return all rows of the right-hand-side table no matter whether there is a match or not

Display **department** info associated with their **student** info. For those departments who does not have any students, leave the student-related columns with NULL values

```
SELECT *
FROM Student RIGHT OUTER JOIN Department ON
     Student.dno = Department.dno;
```

The keyword OUTER can be omitted

Right Outer Join (2)

Sid	Name	Age	dno
1	James	25	1
2	Lucas	24	1
3	Alice	21	2
4	Bruce	30	

dno	Name	Head
1	Management Sciences	Nick
2	Accounting	Kate
3	Computer Science	Alberto

Sid	Name	Age	dno	dno	Name	Head
1	James	25	1	1	Management Sciences	Nick
2	Lucas	24	1	1	Management Sciences	Nick
3	Alice	21	2	2	Accounting	Kate
				3	Computer Science	Alberto

Leave student-related columns with NULL values

Full Outer Join

The keyword OUTER can be omitted

```
SELECT *
FROM Student FULL OUTER JOIN Department ON
     Student.dno = Department.dno;
```

Sid	Name	Age	dno
1	James	25	1
2	Lucas	24	1
3	Alice	21	2
4	Bruce	30	

dno	Name	Head
1	Management Sciences	Nick
2	Accounting	Kate
3	Computer Science	Alberto

Sid	Name	Age	dno	dno	Name	Head
1	James	25	1	1	Management Sciences	Nick
2	Lucas	24	1	1	Management Sciences	Nick
3	Alice	21	2	2	Accounting	Kate
4	Bruce	30				
				3	Computer Science	Alberto

- List all customers associated with their order info, including customers with no order info.

```
SELECT *
FROM Customer_T o LEFT JOIN Order_T c ON c.Customerid = o.Customerid
```

CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTALCODE	ORDERID	CUSTOMERID	ORDERDATE
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871	1001	1	10/21/2010
1	Contemporary Casuals	1355 S Hines Blvd	Gainesville	FL	32601-2871	1010	1	11/05/2010
2	Value Furniture	15145 S.W. 17th St.	Plano	TX	75094-7743	1006	2	10/24/2010
3	Home Furnishings	1900 Allard Ave.	Albany	NY	12209-1125	1005	3	10/24/2010
4	Eastern Furniture	1925 Beltline Rd.	Carteret	NJ	07008-3188	1009	4	11/05/2010
5	Impressions	5585 Westcott Ct.	Sacramento	CA	94206-4056	1004	5	10/22/2010
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	1002	8	10/21/2010
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	1007	11	10/27/2010
12	Battle Creek Furniture	345 Capitol Ave. SW	Battle Creek	MI	49015-3401	1008	12	10/30/2010
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	-	1003	15	10/22/2010

- Show the ID and names of customers who have not placed any orders.

```
SELECT *
FROM Customer_T o LEFT JOIN Order_T c ON c.CustomerID = o.CustomerID
```

```
SELECT CustomerID, CustomerName
FROM Customer_T o LEFT JOIN Order_T c ON c.CustomerID = o.CustomerID
WHERE OrderID IS NULL;
```

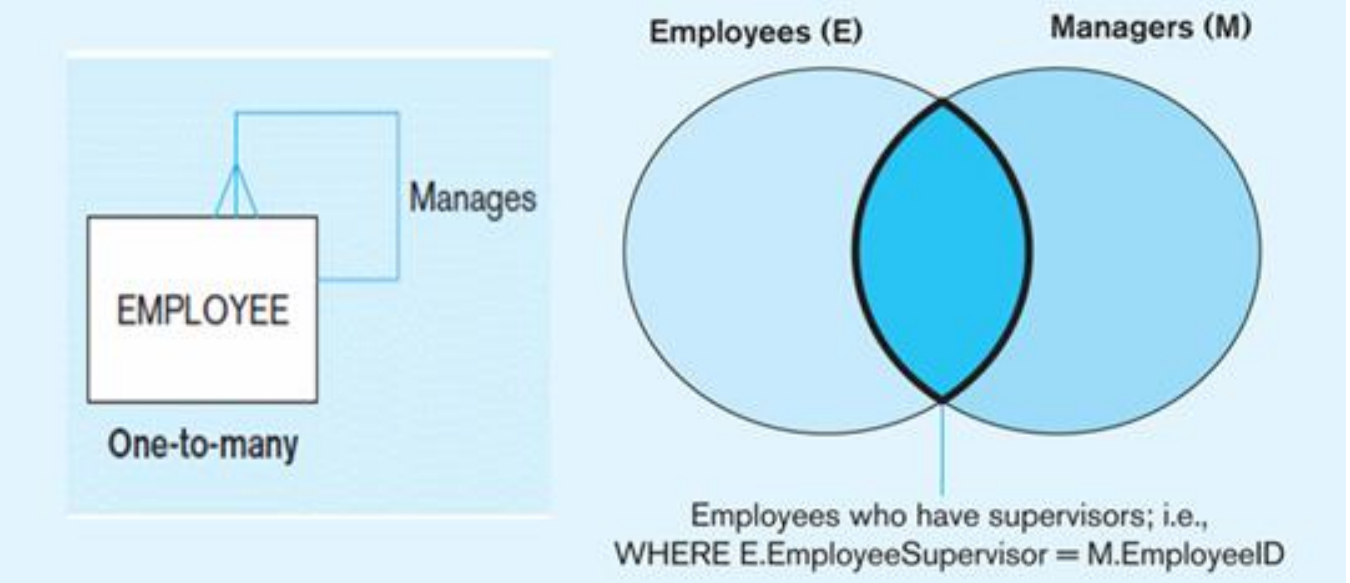
CUSTOMERID	CUSTOMERNAME	CUSTOMERADDRESS	CUSTOMERCITY	CUSTOMERSTATE	CUSTOMERPOSTALCODE	ORDERID	CUSTOMERID	ORDERDATE
6	Furniture Gallery	325 Flatiron Dr.	Boulder	CO	80514-4432	-	-	-
7	Period Furniture	394 Rainbow Dr.	Seattle	WA	97954-5589	-	-	-
9	M and H Casual Furniture	3709 First Street	Clearwater	FL	34620-2314	-	-	-
10	Seminole Interiors	2400 Rocky Point Dr.	Seminole	FL	34646-4423	-	-	-
13	Heritage Furnishings	66789 College Ave.	Carlisle	PA	17013-8834	-	-	-
14	Kaneohe Homes	112 Kiowai St.	Kaneohe	HI	96744-2537	-	-	-

Self-Join (1)

- Connecting rows in the same table in pairs (i.e., joining a table with itself)
- Implement a unary relationship (or used as a special technique)
- Need to rename one of the tables

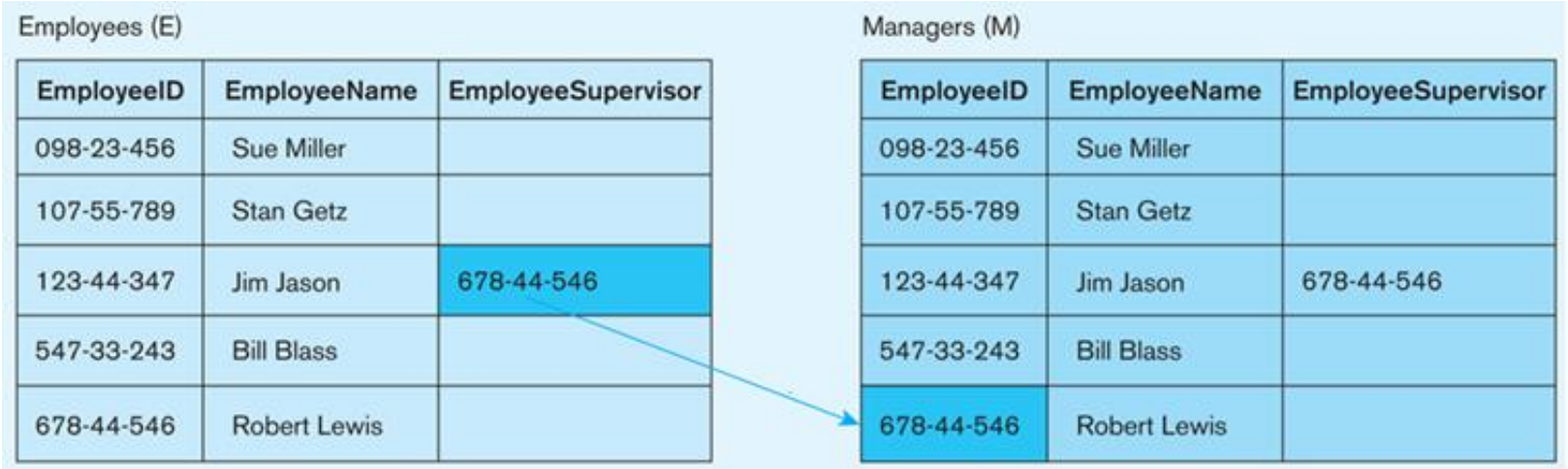
Employee(EmployeeID, EmpployeeName, EmployeeSupervisor)

Query:
find out all the managing relationships, i.e., EmployeeID, EmployeeName, and Supervior's name



Self-Join (2)

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



Result:

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

- For the PVFC dataset, find all pairs of customers in the same state

CUSTOMERID	CUSTOMERNAME	CUSTOMERSTATE
1	Contemporary Casuals	FL
2	Value Furniture	TX
3	Home Furnishings	NY
4	Eastern Furniture	NJ
5	Impressions	CA
6	Furniture Gallery	CO
7	Period Furniture	WA
8	California Classics	CA
9	M and H Casual Furniture	FL
10	Seminole Interiors	FL
11	American Euro Lifestyles	NJ
12	Battle Creek Furniture	MI
13	Heritage Furnishings	PA
14	Kaneohe Homes	HI
15	Mountain Scenes	UT

CUSTOMERID	CUSTOMERNAME	CUSTOMERSTATE
1	Contemporary Casuals	FL
2	Value Furniture	TX
3	Home Furnishings	NY
4	Eastern Furniture	NJ
5	Impressions	CA
6	Furniture Gallery	CO
7	Period Furniture	WA
8	California Classics	CA
9	M and H Casual Furniture	FL
10	Seminole Interiors	FL
11	American Euro Lifestyles	NJ
12	Battle Creek Furniture	MI
13	Heritage Furnishings	PA
14	Kaneohe Homes	HI
15	Mountain Scenes	UT

- For the PVFC dataset, find all pairs of customers in the same state

```
SELECT c1.CustomerName, c2.CustomerName, c1.CustomerState
FROM Customer_T c1 LEFT JOIN Customer_T c2 ON c1.CustomerState = c2.CustomerState;
```

c1.CUSTOMERNAME	c2.CUSTOMERNAME	c1.CUSTOMERSTATE
Contemporary Casuals	Contemporary Casuals	FL
Seminole Interiors	Contemporary Casuals	FL
Contemporary Casuals	Seminole Interiors	FL

- For the PVFC dataset, find all pairs of different customers in the same state. Remove duplicate pairs (AAA, BBB is same as BBB, AAA so only one of them should exist in the result).

```
SELECT c1.CustomerName, c2.CustomerName, c1.CustomerState
FROM Customer_T c1 LEFT JOIN Customer_T c2 ON c1.CustomerState = c2.CustomerState
WHERE c1.CustomerID < c2.CustomerID;
```

Exercise

- Find all the pairs of products that have been purchased in the same order. Show ProductID, of all such pairs. Remove any duplicate pairs.

```
SELECT DISTINCT o1.ProductID, o2.ProductID
FROM Orderline_T o1, Orderline_T o2
WHERE o1.OrderID = o2.OrderID AND o1.ProductID < o2.ProductID
```

Order 1001

Product 1
Product 2
Product 3

Order 1002

Product 1
Product 3
Product 4

O1.ProductID	O2.ProductID
1	2
1	3
2	3
1	3
1	4
3	4

O1.ProductID	O2.ProductID
1	2
1	3
2	3
~~1~~	~~3~~
1	4
3	4