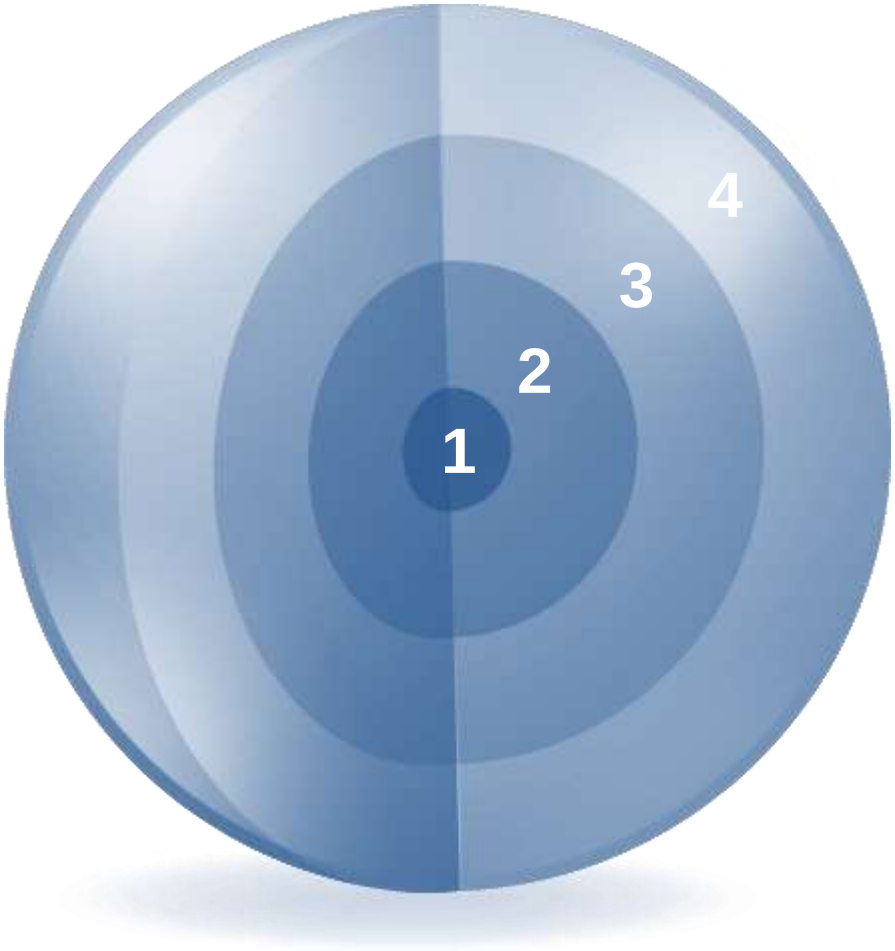
Displaying Data from Multiple Tables Using Joins

What You will learn at the end of this Session?



1. Write SELECT statements to access data from more than one table using equijoins and nonequijoins

2. Join a table to itself by using a self-join

3. View data that generally does not meet a join condition by using OUTER joins

4. Generate a Cartesian product of all rows from two or more tables

Obtaining Data from Multiple Tables

EMPLOYEES

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
...			
18	174	Abel	80
19	176	Taylor	80
20	178	Grant	(null)

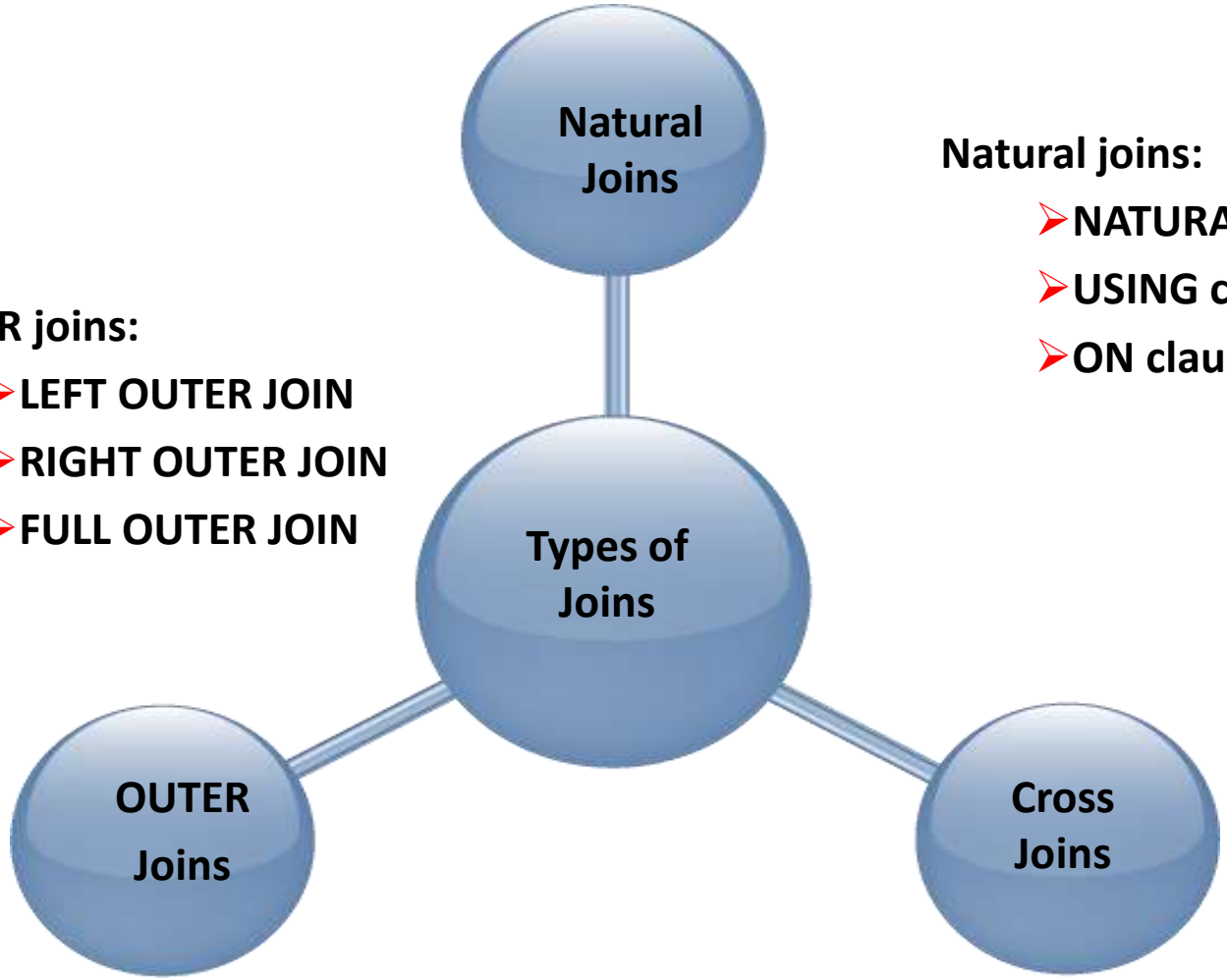
DEPARTMENTS

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

	EMPLOYEE_ID	DEPARTMENT_ID	DEPARTMENT_NAME
1	200	10	Administration
2	201	20	Marketing
3	202	20	Marketing
4	124	50	Shipping
...			
18	205	110	Accounting
19	206	110	Accounting

Types of Joins

Joins that are compliant with the SQL:1999 standard include the following:



Natural joins:

- **NATURAL JOIN clause**
- **USING clause**
- **ON clause**

OUTER joins:

- **LEFT OUTER JOIN**
- **RIGHT OUTER JOIN**
- **FULL OUTER JOIN**

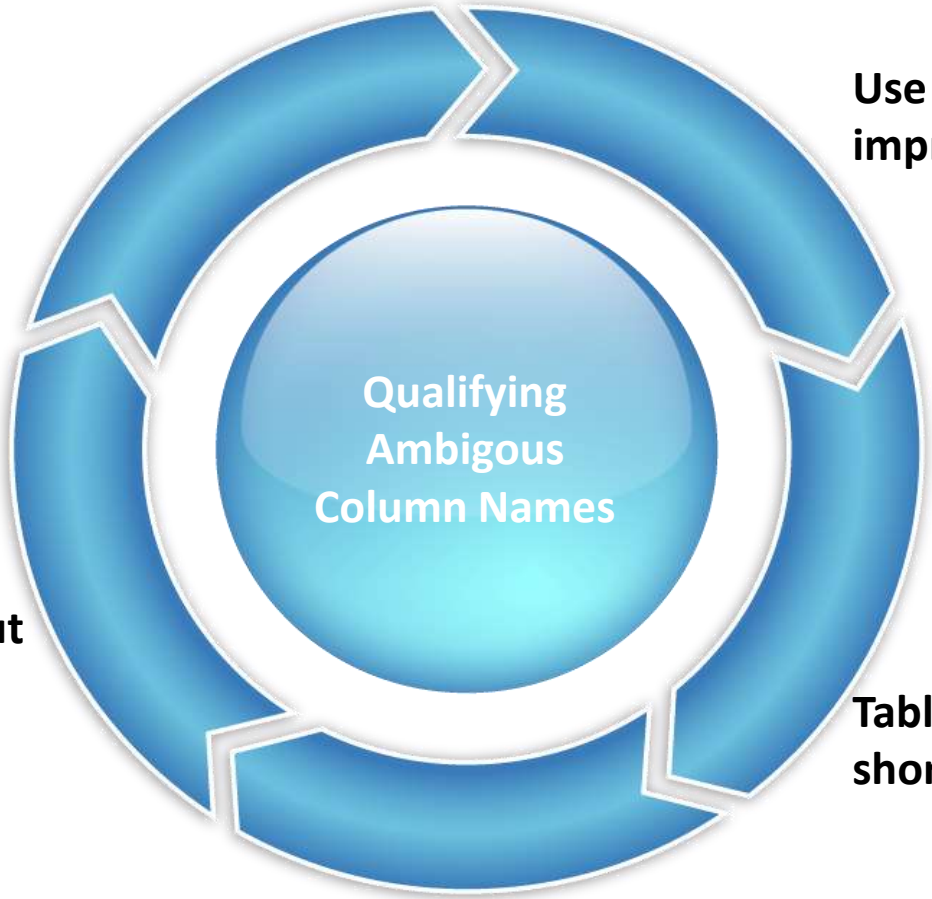
Joining Tables Using SQL:1999 Syntax

Use a join to query data from more than one table:

```
SELECT   table1.column, table2.column
FROM     table1
[NATURAL JOIN table2] |
[JOIN table2 USING (column_name)] |
[JOIN table2
  ON (table1.column_name = table2.column_name)]|
[LEFT|RIGHT|FULL OUTER JOIN table2
  ON (table1.column_name = table2.column_name)]|
[CROSS JOIN table2];
```

Qualifying Ambiguous Column Names

Use table prefixes to qualify column names that are in multiple tables.



Use table prefixes to improve performance.

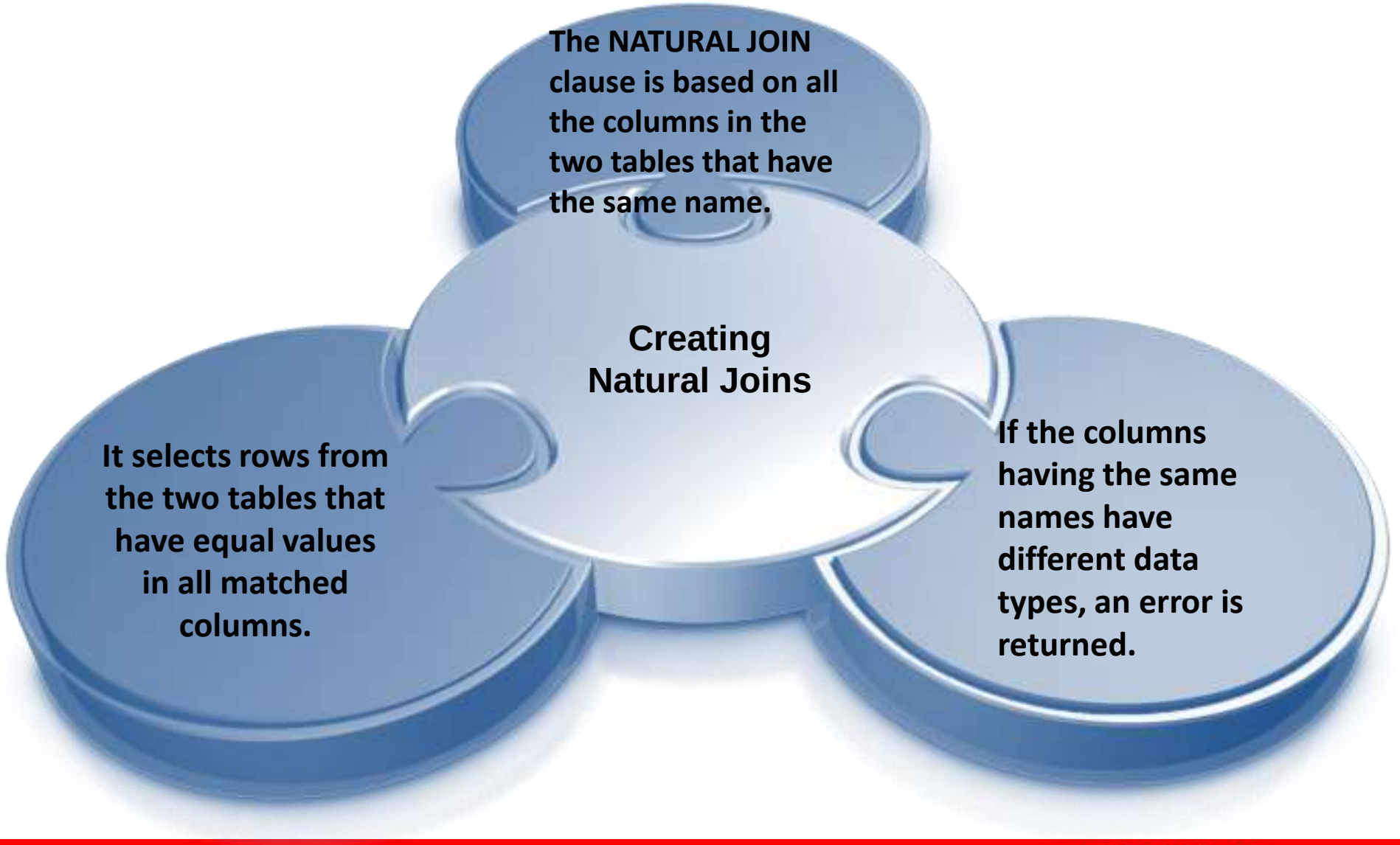
Instead of full table name prefixes, use table aliases.

Table alias gives a table a shorter name:

- **Keeps SQL code smaller, uses less memory**

Use column aliases to distinguish columns that have identical names, but reside in different tables.

Creating Natural Joins



Retrieving Records with Natural Joins

```
SELECT   order_id,  to_char (order_date, ' fmDD Month YYYY ')
AS   "ORDER DATE",  order_status,  customer_id
FROM  orders
NATURAL JOIN  customers ;
```

	ORDER_ID	ORDER DATE	ORDER_STATUS	CUSTOMER_ID
1	2458	17 August 1999	0	101
2	2447	27 July 2000	8	101
3	2413	30 March 2000	5	101
4	2430	2 October 1999	8	101
5	2397	20 November 1999	1	102
6	2432	14 September 1999	10	102
7	2414	30 March 1999	8	102
8	2431	14 September 1998	1	102
9	2454	3 October 1999	1	103
10	2437	1 September 1998	4	103
11	2433	13 September 2099	10	103
12	2415	29 March 2097	6	103

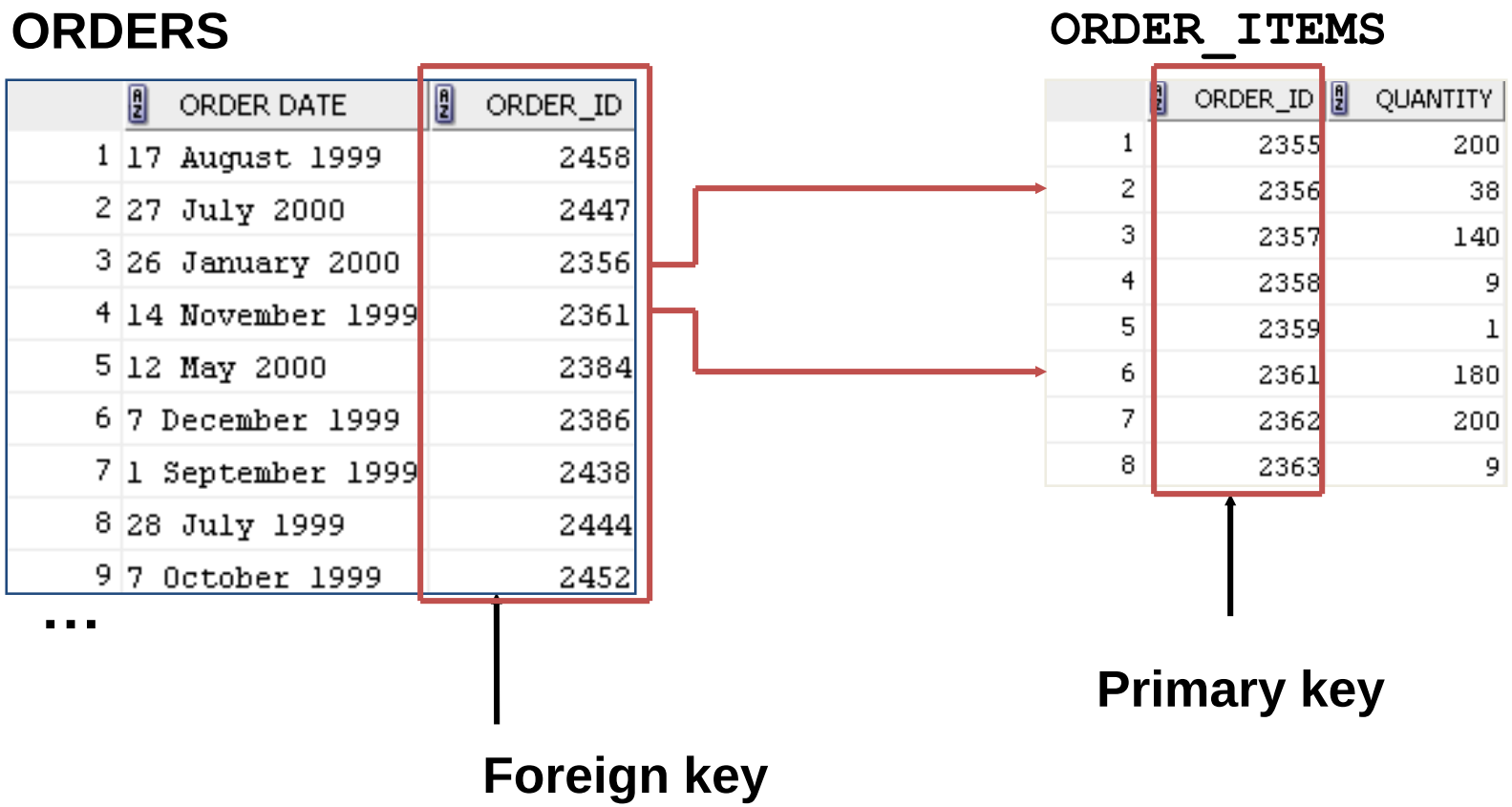
Creating Joins with the USING Clauseb

If several columns have the same names but the data types do not match, use the USING clause to specify the columns for the equijoin.

Use the USING clause to match only one column when more than one column matches.

The NATURAL JOIN and USING clauses are mutually exclusive.

Joining Column Names



Retrieving Records with the USING Clause

```
SELECT  order_id, order_status, customer_id, cust_first_name
FROM orders  JOIN  customers
USING  (customer_id) ;
```

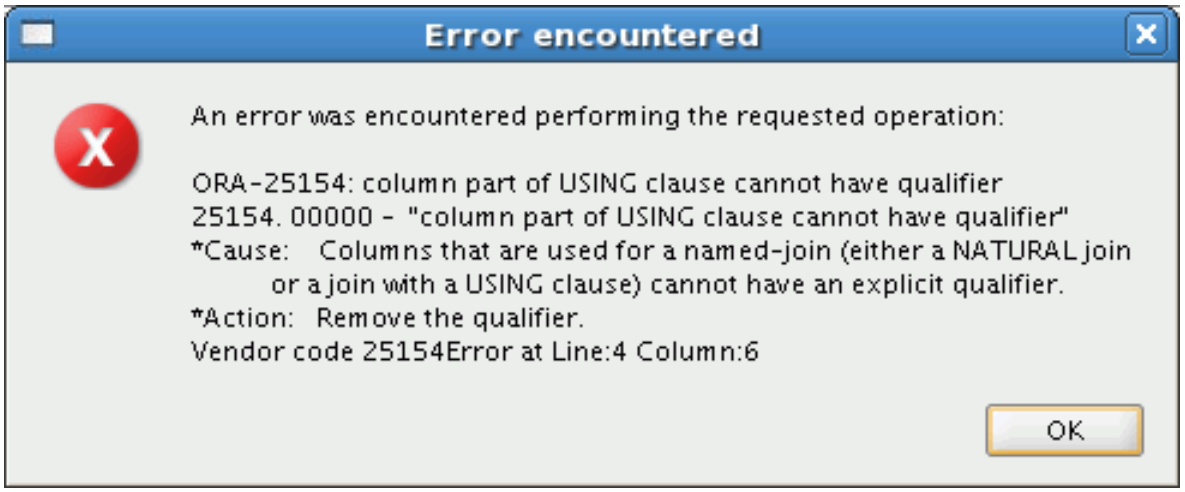
	ORDER_ID	ORDER_STATUS	CUSTOMER_ID	CUST_FIRST_NAME
1	2458	0	101	Constantin
2	2447	8	101	Constantin
3	2413	5	101	Constantin
4	2430	8	101	Constantin
5	2397	1	102	Harrison
6	2432	10	102	Harrison
7	2414	8	102	Harrison
8	2431	1	102	Harrison
...				
64	2448	5	145	Mammutti
65	2379	8	146	Elia

Using Table Aliases with the USING Clause

Do not qualify a column that is used in the USING clause.

If the same column is used elsewhere in the SQL statement, do not alias it.

```
SELECT l.city, d.department_name
FROM   locations l JOIN departments d
USING (location_id)
WHERE d.location_id = 1400;
```



Creating Joins with the ON Clause

The join condition for the natural join is basically an equijoin of all columns with the same name.

Use the ON clause to specify arbitrary conditions or specify columns to join.

The join condition is separated from other search conditions.

The ON clause makes code easy to understand.

Retrieving Records with the ON Clause

```
SELECT   e.order_status,  e.customer_id,  e.order_id,
         d.order_id,  d.quantity
FROM   orders e  JOIN  order_items d
ON   (e.order_id = d.order_id) ;
```

	ORDER_STATUS	CUSTOMER_ID	ORDER_ID	ORDER_ID_1	QUANTITY
1	8	104	2355	2355	200
2	5	105	2356	2356	38
3	5	108	2357	2357	140
4	2	105	2358	2358	9
5	9	106	2359	2359	1
6	8	108	2361	2361	180
7	4	109	2362	2362	200
8	0	144	2363	2363	9
9	4	145	2364	2364	6

...

Creating Three-Way Joins with the ON Clause

```
SELECT   customer_id,  unit_price,  warehouse_id
FROM   orders e
JOIN   order_items d
ON   e.order_id  =  d.order_id
JOIN   inventories f
ON   d.product_id  =  f.product_id;
```

	CUSTOMER_ID	UNIT_PRICE	WAREHOUSE_ID
1	105	199.1	9
2	105	199.1	2
3	105	199.1	4
4	105	199.1	6
5	105	199.1	8
6	105	226.6	9
7	105	226.6	2
8	105	226.6	4
9	105	226.6	6
10	105	226.6	8
11	106	270.6	6
12	144	199.1	9

...

Applying Additional Conditions to a Join

Use the AND clause or the WHERE clause to apply additional conditions:

```
SELECT   e.order_status, e.customer_id, e.order_id,
         d.order_id,  d.quantity
FROM   orders e  JOIN  order_items d
ON    (e.order_id = d.order_id)
AND e.order_status = 0;
```

Or

```
SELECT   e.order_status, e.customer_id, e.order_id,
         d.order_id,  d.quantity
FROM   orders e  JOIN  order_items d
ON    (e.order_id = d.order_id)
WHERE e.order_status = 0;
```

Joining a Table to Itself

EMPLOYEES (WORKER)

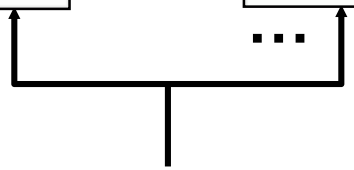
EMPLOYEE_ID	LAST_NAME	MANAGER_ID
200	Whalen	101
201	Hartstein	100
202	Fay	201
205	Higgins	101
206	Gietz	205
100	King	(null)
101	Kochhar	100
102	De Haan	100
103	Hunold	102
104	Ernst	103

...

EMPLOYEES (MANAGER)

EMPLOYEE_ID	LAST_NAME
200	Whalen
201	Hartstein
202	Fay
205	Higgins
206	Gietz
100	King
101	Kochhar
102	De Haan
103	Hunold
104	Ernst

...



MANAGER_ID in the WORKER table is equal to EMPLOYEE_ID in the MANAGER table.

Self-Joins Using the ON Clause

```
SELECT worker.last_name emp, manager.last_name mgr
FROM   employees worker JOIN employees manager
ON     (worker.manager_id = manager.employee_id);
```

	EMP	MGR
1	Hunold	De Haan
2	Fay	Hartstein
3	Gietz	Higgins
4	Lorentz	Hunold
5	Ernst	Hunold
6	Zlotkey	King
7	Mourgos	King

...

EMPLOYEES

	LAST_NAME	SALARY
1	Whalen	4400
2	Hartstein	13000
3	Fay	6000
4	Higgins	12000
5	Gietz	8300
6	King	24000
7	Kochhar	17000
8	De Haan	17000
9	Hunold	9000
10	Ernst	6000
...		
19	Taylor	8600
20	Grant	7000

JOB_GRADES

	GRADE_LEVEL	LOWEST_SAL	HIGHEST_SAL
1	A	1000	2999
2	B	3000	5999
3	C	6000	9999
4	D	10000	14999
5	E	15000	24999
6	F	25000	40000

The JOB_GRADES table defines the LOWEST_SAL and HIGHEST_SAL range of values for each GRADE_LEVEL. Therefore, the GRADE_LEVEL column can be used to assign grades to each employee.

Retrieving Records with Nonequijoins

```
SELECT e.last_name, e.salary, j.grade_level
FROM   employees e JOIN job_grades j
ON     e.salary
       BETWEEN j.lowest_sal AND j.highest_sal;
```

	LAST_NAME	SALARY	GRADE_LEVEL
1	Vargas	2500	A
2	Matos	2600	A
3	Davies	3100	B
4	Rajs	3500	B
5	Lorentz	4200	B
6	Whalen	4400	B
7	Mourgos	5800	B
8	Ernst	6000	C
9	Fay	6000	C
10	Grant	7000	C

...

Returning Records with No Direct Match Using OUTER Joins

DEPARTMENTS

	DEPARTMENT_NAME	DEPARTMENT_ID
1	Administration	10
2	Marketing	20
3	Shipping	50
4	IT	60
5	Sales	80
6	Executive	90
7	Accounting	110
8	Contracting	190

There are no employees in department 190.

Equijoin with EMPLOYEES

	DEPARTMENT_ID	LAST_NAME
1	10	Whalen
2	20	Hartstein
3	20	Fay
4	110	Higgins
5	110	Gietz
6	90	King
7	90	Kochhar
8	90	De Haan
9	60	Hunold
10	60	Ernst
...		
18	80	Abel
19	80	Taylor

Employee "Grant" has not been assigned a department ID.

INNER Versus OUTER Joins

In SQL:1999, the join of two tables returning only matched rows is called an INNER join.

A join between two tables that returns the results of the INNER join as well as the unmatched rows from the left (or right) table is called a left (or right) OUTER join.

A join between two tables that returns the results of an INNER join as well as the results of a left and right join is a full OUTER join.

LEFT OUTER JOIN

```
SELECT e.last_name, e.department_id, d.department_name
FROM   employees e LEFT OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Fay	20	Marketing
3	Hartstein	20	Marketing
4	Vargas	50	Shipping
5	Matos	50	Shipping

...

16	Kochhar	90	Executive
17	King	90	Executive
18	Gietz	110	Accounting
19	Higgins	110	Accounting
20	Grant	(null)	(null)

RIGHT OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e RIGHT OUTER JOIN departments d
ON     (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Davies	50	Shipping
5	Vargas	50	Shipping
6	Rajs	50	Shipping
7	Mourgos	50	Shipping
8	Matos	50	Shipping

...

18	Higgins	110	Accounting
19	Gietz	110	Accounting
20	(null)	190	Contracting

FULL OUTER JOIN

```
SELECT e.last_name, d.department_id, d.department_name
FROM   employees e FULL OUTER JOIN departments d
ON    (e.department_id = d.department_id) ;
```

	LAST_NAME	DEPARTMENT_ID	DEPARTMENT_NAME
1	Whalen	10	Administration
2	Hartstein	20	Marketing
3	Fay	20	Marketing
4	Higgins	110	Accounting

...

17	Zlotkey	80	Sales
18	Abel	80	Sales
19	Taylor	80	Sales
20	Grant	(null)	(null)
21	(null)	190	Contracting

Cartesian Products

A Cartesian product is formed when:

- A join condition is omitted
- A join condition is invalid
- All rows in the first table are joined to all rows in the second table

Always include a valid join condition if you want to avoid a Cartesian product.

Generating a Cartesian Product

EMPLOYEES (20 rows)

	EMPLOYEE_ID	LAST_NAME	DEPARTMENT_ID
1	200	Whalen	10
2	201	Hartstein	20
3	202	Fay	20
4	205	Higgins	110
...			
19	176	Taylor	80
20	178	Grant	(null)

DEPARTMENTS (8 rows)

	DEPARTMENT_ID	DEPARTMENT_NAME	LOCATION_ID
1	10	Administration	1700
2	20	Marketing	1800
3	50	Shipping	1500
4	60	IT	1400
5	80	Sales	2500
6	90	Executive	1700
7	110	Accounting	1700
8	190	Contracting	1700

Cartesian product: 20 x 8 = 160 rows

	EMPLOYEE_ID	DEPARTMENT_ID	LOCATION_ID
1	200	10	1700
2	201	20	1700
...			
21	200	10	1800
22	201	20	1800
...			
159	176	80	1700
160	178	(null)	1700

Creating Cross Joins

- The CROSS JOIN clause produces the cross-product of two tables.
- This is also called a Cartesian product between the two tables.

```
SELECT last_name, department_name
FROM   employees
CROSS JOIN departments ;
```

	LAST_NAME	DEPARTMENT_NAME
1	Abel	Administration
2	Davies	Administration
3	De Haan	Administration
4	Ernst	Administration
5	Fay	Administration

...

158	Vargas	Contracting
159	Whalen	Contracting
160	Zlotkey	Contracting

Historical Joins 8i

Equi join

Department Table

DEPT_CODE	DEPT_HEAD
IMG	7499
BSFI	6348
TMTS	7698
NEW1	
NEW2	

EmployeesTable

EMPNO	NAME	DEPT_CODE	LOC_CODE
7499	RAM	IMG	BDC
7369	GOPAL	BSFI	BDC
7698	NAREN	TMTS	CDC
6348	VIVEK	BSFI	CDC
7021	JOSEPH	IMG	PDC
7688	RAHEEM	IMG	HDC

Joining Purpose: to List out Department heads and their names

Joining Condition : Department.dept_Head = Employees.EMPNO

Equi join (Contd.).

Cartesian Product

DEPT_HEAD	EMPNO
7499	7499
7499	7369
7499	7698
7499	6348
7499	7021
7499	7688
6348	7499
6348	7369
6348	7698
6348	6348
6348	7021
6348	7688
7698	7499
7698	7369
7698	7698
7698	6348
7698	7021
7698	7688

Equi join Result

DEPT_CODE	DEPT_HEAD	NAME
IMG	7499	RAM
TMTS	7698	NAREN
BSFI	6348	VIVEK

Non Equi join

Income Tax (Alias T)

Low_Ann_Sal	High_Ann_Sal	IT_Slab
10000	12000	1
12001	16000	2
16001	22000	3
22001	99999	4

EmployeesTable (Alias E)

EMPNO	NAME	DEPT_CODE	Ann_SAL
7499	RAM	IMG	12000
7369	GOPAL	BSFI	14000
7698	NAREN	TMTS	17000
6348	VIVEK	BSFI	12000
7021	JOSEPH	IMG	15000
7688	RAHEEM	IMG	28000

Joining Purpose: to List out Employees and their respective Income Tax Slab

Joining Condition: E.Ann_Sal T.Low_Ann_Sal <= E.Ann_Sal T.High_Ann_Sal

Non Equi join Result

EMPNO	ENAME	DEPT_CODE	Ann_SAL
7499	RAM	IMG	12000
7369	GOPAL	BSFI	14000
7698	NAREN	TMTS	17000
6348	VIVEK	BSFI	12000
7021	JOSEPH	IMG	15000
7688	RAHEEM	IMG	28000

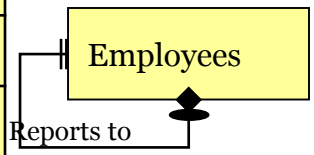
Low_Ann_Sal	High_Ann_Sal	IT_Slab
10000	12000	1
12001	16000	2
16001	22000	3
10000	12000	1
12001	16000	2
22001	99999	4

Select Statement

```
SELECT EMPNO,ENAME,DEPT_CODE,ANN_SAL,Low_Ann_Sal,High_Ann_Sal,IT_SLAB
FROM EMPLOYEES E , ITAX_SLAB T
WHERE E.ANN_SAL BETWEEN T.LOW_ANN_SAL AND T.HIGH_ANN_SAL
```

Self Join

EMPNO	ENAME	JOB	DEPTNO	HIREDATE	MGR	SAL	COMM
7788	ARUN	ANALYST	10	19-Apr-87	7566	2200	
7844	TURNER	SALESMAN	10	8-Sep-81	7698	3000	0
7934	MILLER	CLERK	20	23-Jan-82	7782	2500	
7900	CAPTAIN	CLERK	20	3-Dec-81	7698	2100	
7654	RAVI	SALESMAN	20	28-Sep-81	7698	2500	1400
7839	RAJ	PRESIDENT	20	17-Nov-81		1200	
7782	JOSEPH	MANAGER	30	9-Jun-81	7839	2500	
7566	RAGHU	MANAGER	30	2-Apr-81	7839	900	
7902	CHANDRAN	ANALYST	30	3-Dec-81	7566	2200	
7499	RAM	SALESMAN	30	20-Feb-81	7698	2100	300
7876	AKBAR	CLERK	30	23-May-87	7788	2100	
7698	ARJUN	MANAGER	30	1-May-81	7839	1600	
7521	SHYAM	SALESMAN	30	22-Feb-81	7698	800	500



Joining Purpose: to List out Employee details empno,ename,Job,deptno,hiredate,mgr, Employee's Manager's Name (who is also one among the Employees).

Self Join Result

EMPNO	ENAME	JOB	DEPTNO	HIREDATE	MGR	MANAGER
7788	ARUN	ANALYST	10	19-Apr-87	7566	RAGHU
7844	TURNER	SALESMAN	10	8-Sep-81	7698	ARJUN
7934	MILLER	CLERK	20	23-Jan-82	7782	JOSEPH
7900	CAPTAIN	CLERK	20	3-Dec-81	7698	ARJUN
7654	RAVI	SALESMAN	20	28-Sep-81	7698	ARJUN
7839	RAJ	PRESIDENT	20	17-Nov-81		
7782	JOSEPH	MANAGER	30	9-Jun-81	7839	RAJ
7566	RAGHU	MANAGER	30	2-Apr-81	7839	RAJ
7902	CHANDRAN	ANALYST	30	3-Dec-81	7566	RAGHU
7499	RAM	SALESMAN	30	20-Feb-81	7698	ARJUN
7876	AKBAR	CLERK	30	23-May-87	7788	ARUN
7698	ARJUN	MANAGER	30	1-May-81	7839	RAJ
7521	SHYAM	SALESMAN	30	22-Feb-81	7698	ARJUN

Employees table has been imitated as two different tables to be joined.

MGR column assumes the same domain of values as that of EMPNO Column.

Select Statement

```
SELECT E.EMPNO,E.ENAME,E.JOB, E.DEPTNO,E.HIREDATE,E.MGR,M.ENAME as MANAGER
FROM EMPLOYEES E , EMPLOYEES M
WHERE E.MGR = M.EMPNO
```

Left Outer Join

Department Table

DEPT_CODE	DEPT_HEAD
IMG	7499
BSFI	6348
TMTS	7698
NEW1	
NEW2	

EmployeesTable

EMPNO	NAME	DEPT_CODE	LOC_CODE
7499	RAM	IMG	BDC
7369	GOPAL	BSFI	BDC
7698	NAREN	TMTS	CDC
6348	VIVEK	BSFI	CDC
7021	JOSEPH	IMG	PDC
7688	RAHEEM	IMG	HDC

Joining Purpose: List department wise employee details, including the departments without any employees in it too.

Left Outer Join Result

Dept Left outer join Employees

DEPT_CODE	EMPNO	NAME	LOC_CODE
BSFI	7369	GOPAL	BDC
BSFI	6348	VIVEK	CDC
IMG	7499	RAM	BDC
IMG	7021	JOSEPH	PDC
IMG	7688	RAHEEM	HDC
NEW1	Null	Null	Null
NEW2	Null	Null	Null
TMTS	7698	NAREN	CDC

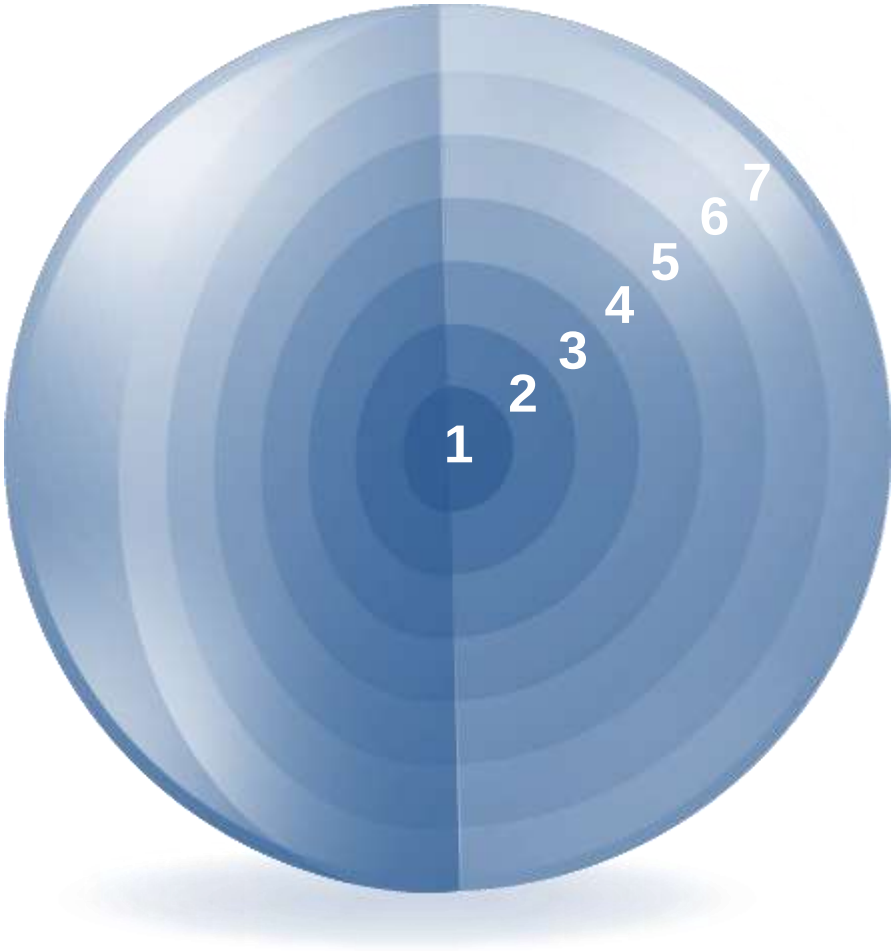

```
Select D.Dept_Code,Empno,Name,Loc_Code
from Dept Left Outer join Employees E
on D.dept_code = E.dept_code
Order by D.Dept_Code
```

Quiz

The SQL:1999 standard join syntax supports the following types of joins. Which of these join types does Oracle join syntax support?

1. Equijoins
2. Nonequijoins
3. Left OUTER join
4. Right OUTER join
5. Full OUTER join
6. Self joins
7. Natural joins
8. Cartesian products

What did you learn at the end of this lesson?



1.Equijoins

2.Nonequijoins

3.OUTER joins

4.Self-joins

5.Cross joins

6.Natural joins

7.Full (or two-sided) OUTER joins

Practice 6: Overview