Cognizant | Academy
Business Focused. Learner Centric.

Oracle
Version 12c

Displaying Data from Multiple Table

Enabling Objectives

After completing this chapter, in the next 120 minutes you will be able to :

Retrieve Data from Multiple Tables by Writing SELECT statements to

- Access data from more than one table using equijoins and non-equijoins
- Join a table to itself by using a self-join
- View data that generally does not meet a join condition by using outer joins

Joins

What is a join?

- A **join** is a query that combines rows from two or more
    - tables,
    - views, or
    - materialized views

  and displays the result to the user.

- The query's select list can select any columns from any of these tables being joined.

- Tables can be joined with each other with the primary key and foreign keys.

Joins

Join Condition

- Is specified in the **Where** clause that compares two columns each from a different table.

Example**:**

If there are two tables **T1** and **T2** with Columns **C1** & **C2** respectively. The join condition will be depicted as

T1.C1=T2.C2

Joins

Execution of Joins of two tables:

- To execute a join, Oracle database combines pairs of rows, each pair containing one row from each table.

- And evaluates the join condition (T1.C1=T2.C2) evaluates to TRUE for the pair, the row is selected for display.

Execution of join of three or more tables:

Assume that there are three tables **T1,T2,T3** with columns **C1,C2,C3** respectively, the join condition will be depicted as

T1.C1=T2.C2 **and** T2.C2=T3.C3

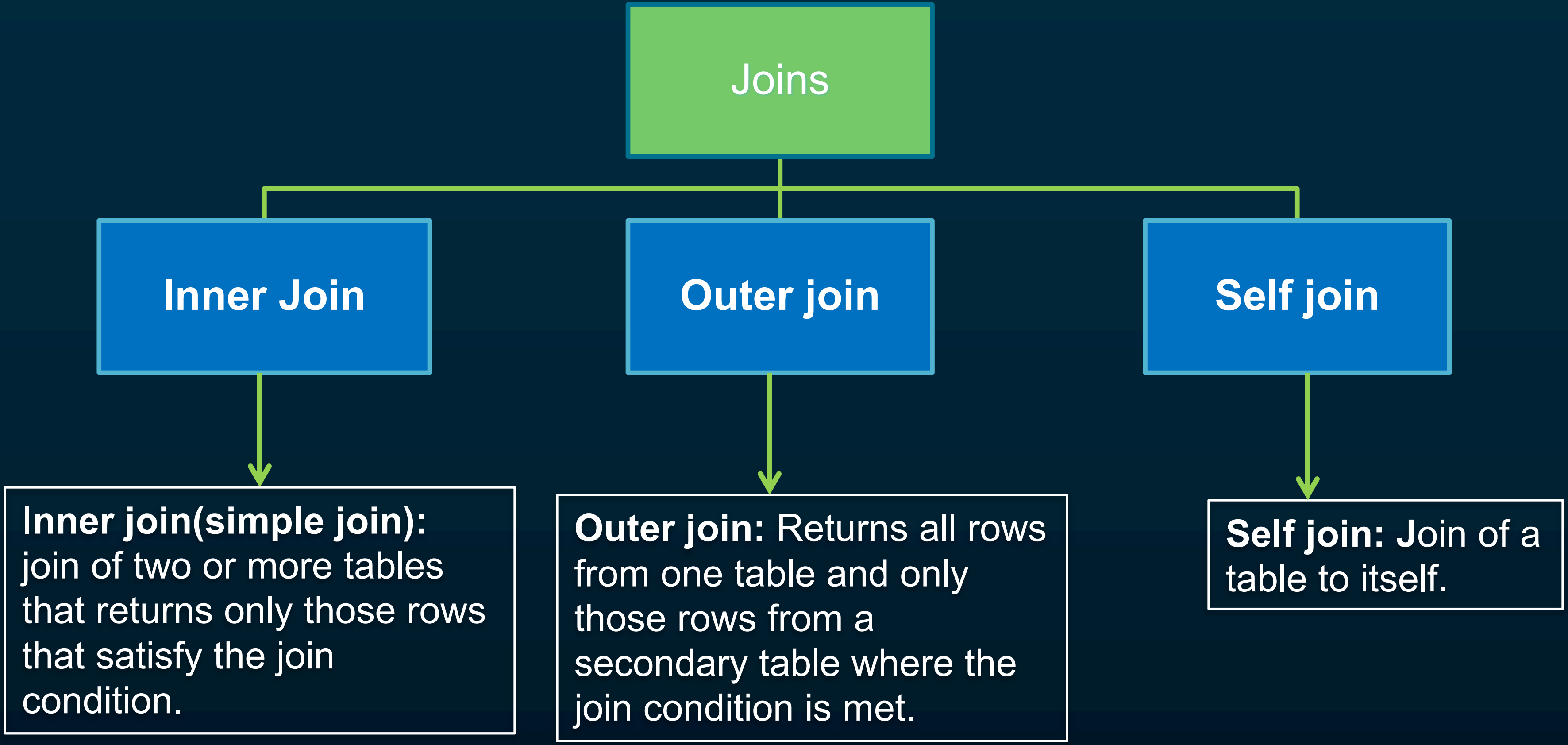
- first joins two of the tables (T1 & T2) based on the join conditions (C1=C2) T1.C1=T2.C2.
- In the retrieved records the next join condition will be fired

T2.C2=T3.C3

Oracle continues this process until all conditions are executed in the query.

Types of Joins

Types of Joins

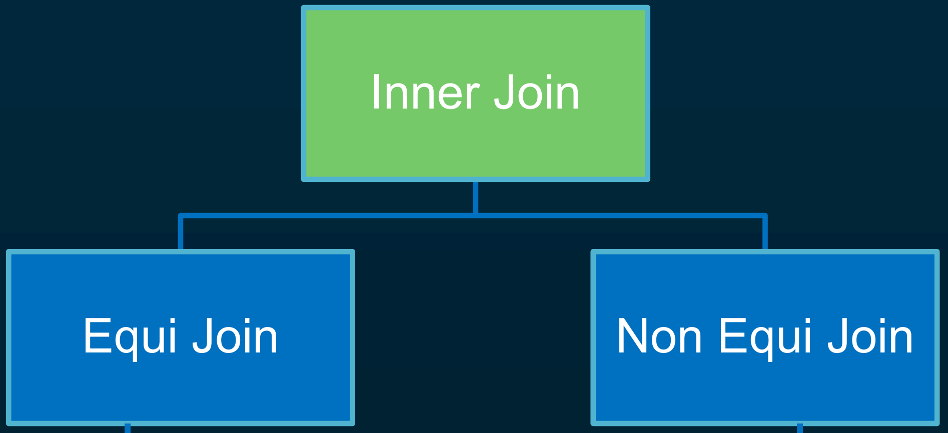


Inner join(simple join): join of two or more tables that returns only those rows that satisfy the join condition.

Outer join: Returns all rows from one table and only those rows from a secondary table where the join condition is met.

Self join: Join of a table to itself.

Types of Inner joins



An **equi join** is a join with a join condition containing an equality operator.

An **non equi join** is a join with a join condition containing an operator other than the equality operator.

Equi Join

Equi-join

- more than one tables are joined together with the help of a common column that exists in both the tables.
- = operator is used to relate the rows of two tables.

Syntax:

SELECT column_name(s)
FROM table_name1,table_name2
WHERE $\underbrace{\text{table\_name1.column\_name=table\_name2.column\_name}}_{\text{Join condition}}$

Equi Join Example

Let us consider an example

- We have a table **PersonsInfo** which stores a persons information like **Persons Id** ,**Last name** , **First name** , **Address** and **City**
- We have another table **OrdersInfo** which stores information like the orders a person has placed like **OrderId**, **Order number** and **Persons Id**

Query for retrieving the orders belonging to all the persons:

```
SELECT P.LastName, P.FirstName, O.OrderNo
FROM PersonsInfo , P,OrdersInfo O
WHERE P.P_Id=O.P_Id;
```

Result**:** The above query retrieves all the records where the person id's match in both the tables.

Equi Join Example

PersonsInfo

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

OrdersInfo

P_Id	O_Id	OrderNo
3	1	77895
2	2	44678
1	3	22456
1	4	24562
15	5	34764

Result:

LastName	FirstName	OrderNo
Hansen	Ola	22456
Hansen	Ola	24562
Svendson	Tove	44678
Pettersen	Kari	77895

Only the highlighted rows are selected since they satisfy the join condition that is person id are same in both the tables.

Lend A Hand - Prerequisites

Pre-requisite # 1 : Associates should ensure that the below tables are available in the oracle database.

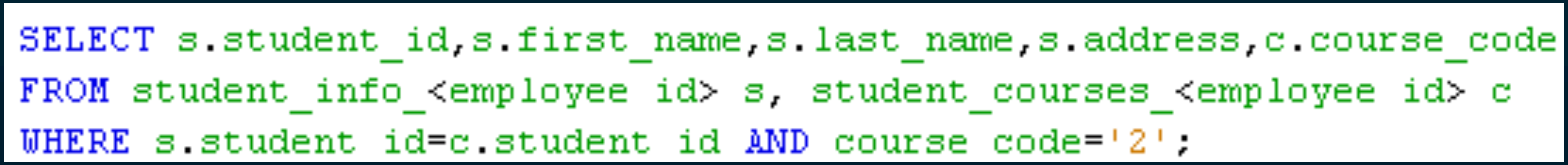
Pre-requisite # 2: Load the table with data using the DML statements.

Lend a Hand – Equi Join

Problem Statement:

Write a query to fetch details of the students who have enrolled for course whose course_code is 2. Student_Info and student_courses to be queried.

Solution:



```
SELECT s.student_id,s.first_name,s.last_name,s.address,c.course_code
FROM student_info_<employee id> s, student_courses_<employee id> c
WHERE s.student id=c.student id AND course code='2';
```

Output:

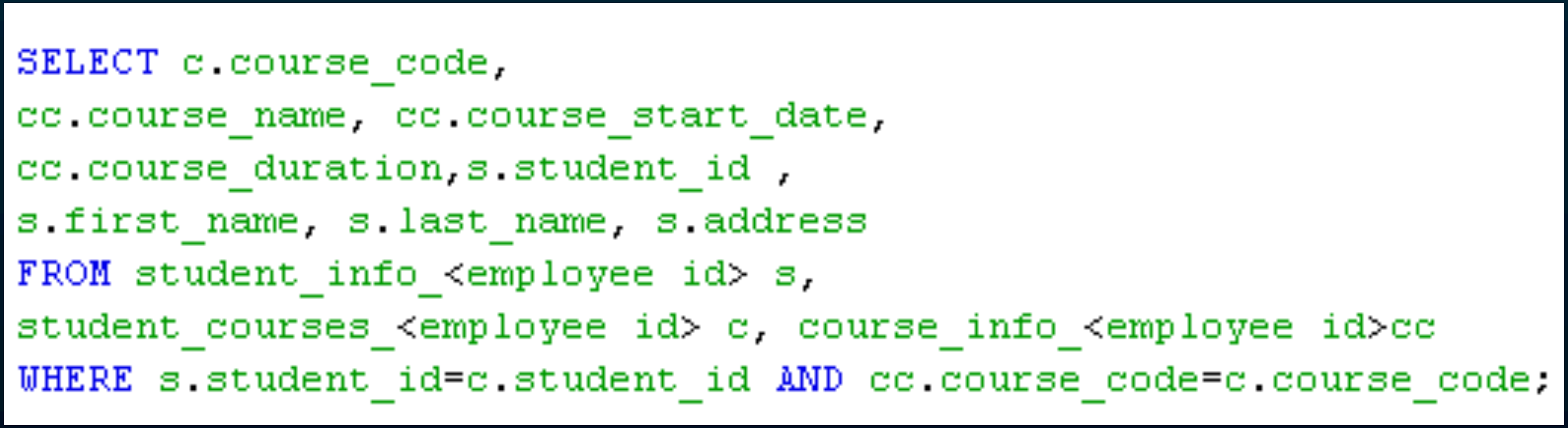
STUDENT_ID	FIRST_NAME	LAST_NAME	ADDRESS	COURSE_CODE
MC02	Simon	Thomson	Meadow Lakes,Sterling	2
MC12	Andrew	Lewis	Fredonia,Fountain Hills	2

Lend a Hand – Equi Join

Problem Statement:

Write a query to fetch details of all the students who have enrolled for course as well as the details of the courses which they have enrolled. student_info, student_courses and course_info to be queried.

Solution:



```
SELECT c.course_code,
cc.course_name, cc.course_start_date,
cc.course_duration,s.student_id ,
s.first_name, s.last_name, s.address
FROM student_info_<employee id> s,
student_courses_<employee id> c, course_info_<employee id>cc
WHERE s.student_id=c.student_id AND cc.course_code=c.course_code;
```

Lend a Hand – Equi Join

Output:

COURSE_CODE	COURSE_NAME	COURSE_START_DATE	COURSE_DURATION	STUDENT_ID	FIRST_NAME	LAST_NAME	ADDRESS
1	Java Programming	2012-01-12	5	MC01	James	Watson	Eielson AFB,Alaska
1	Java Programming	2012-01-12	5	MC11	Scott	Walter	Camp Verde,Coolidge
………	………..	……..	…….	………	……..	…….	…….
6	Cobol Programming	2012-01-18	5	MC23	Paul	Walker	Bristol,Chattahoochee Hills
6	Cobol Programming	2012-01-18	5	MS11	Lara	Soene	Chickamauga,Country Club Estat

Non-Equi Join

Non-equi join

- Is defined as a join in which more than one tables are joined together where there is no direct correspondence between columns in the tables.
- We use operators such as <=,>=, <>, BETWEEN to relate the rows of two tables.

Syntax:

SELECT column_name(s)
FROM table_name1 , table_name2
WHERE $\underbrace{\text{table\_name1.column\_name >=table\_name2.column\_name}}_{\text{Join condition}}$

Non-Equi Join Example

Let us consider an example,

- We have a table **PersonsInfo** which stores information like **Persons Id** ,**Last name , First name , Address** and **Salary.**
- We have another table **GradeInfo** which stores information like **Grade**, **Low Salary** and **High Salary**.

Query to retrieve the grade of all the persons:

```
SELECT P.LastName,P.FirstName,O.Grade
FROM PersonsInfo P, OrdersInfo O
WHERE P.Salary between O.Lowsalary and O.Highsalary;
```

Output**:**

This retrieves all the records from **PersonsInfo** along with the grade information from the **GradeInfo** table.

Non-Equi Join Example

PersonsInfo:

P_Id	LastName	FirstName	Address	Salary
1	Hansen	Ola	Timoteivn 10	1000
2	Svendson	Tove	Borgvn 23	2000
3	Pettersen	Kari	Storgt 20	3000

GradeInfo

Grade	Low salary	High salary
1	1000	2000
2	2001	3000
3	3001	5000
4	5001	10000

Output:

LastName	FirstName	Grade
Hansen	Ola	1
Svendson	Tove	1
Pettersen	Kari	2

The grade for all the persons is been selected based on the join condition P.salary between O.lowsalary and O.highsalary

Lend a Hand – Non Equi Join

Pre-requisite:

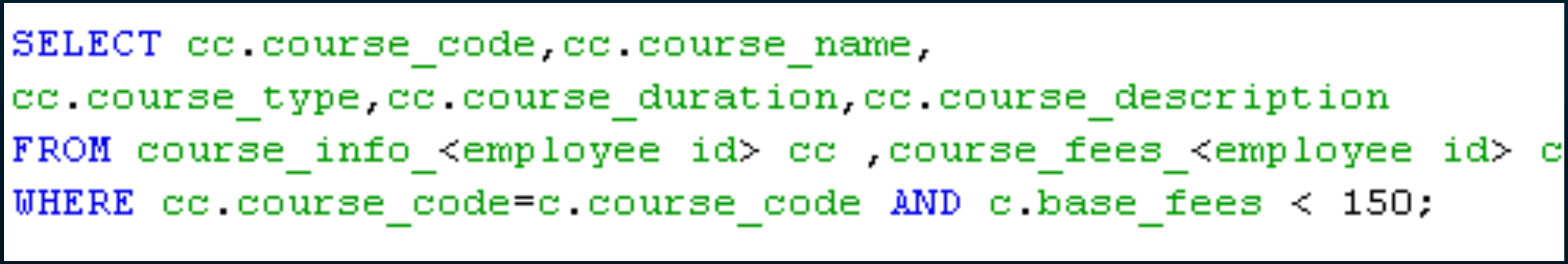
Create two courses (Courses_Info_<Employee Id>) and also add fees < 150 (courses_fees_<Employee Id>).

Lend a Hand – Non Equi Join

Problem # 1: Write a query to fetch the course details namely code, name, course type, duration, description whose base fees is less than 150.

Expected output: The course details added should be retrieved.

Solution # 1:



```
SELECT cc.course_code,cc.course_name,
cc.course_type,cc.course_duration,cc.course_description
FROM course_info_<employee id> cc ,course_fees_<employee id> c
WHERE cc.course_code=c.course_code AND c.base_fees < 150;
```

Lend a Hand Solution – Non Equi Join

Problem # 2:

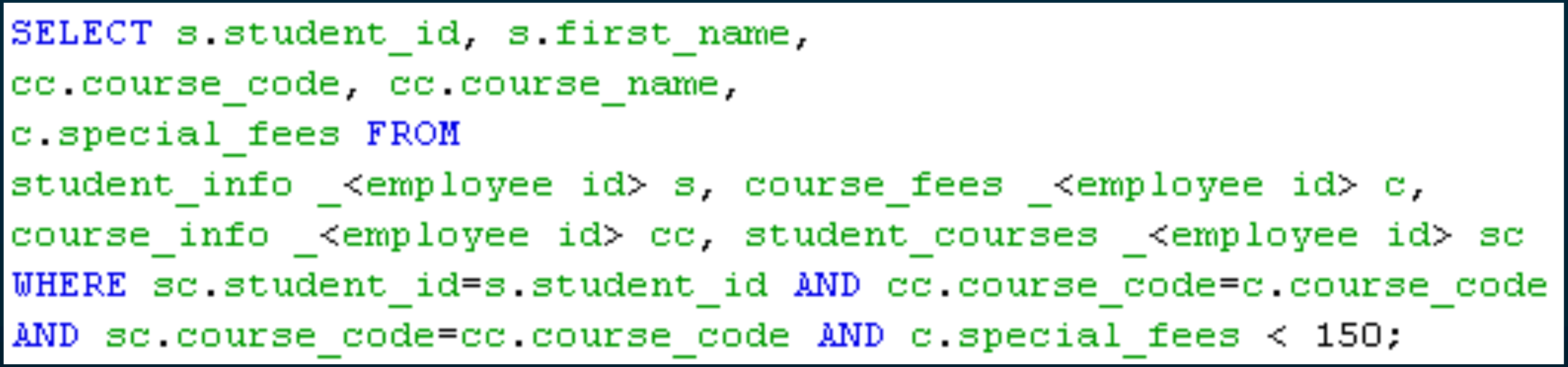
Write a query to fetch details of the student and course details for which base fees is less than 150.

Joining four tables Student_Info_<Employee Id>, Course_Fees_<Emplyee_Id>,Course_Info_<Emplyee_Id>, Student_courses_<Emplyee_Id>.

Expected output: The course and student details added should be retrieved.

Lend a Hand Solution – Non Equi Join

Solution # 2:



```
SELECT s.student_id, s.first_name,
cc.course_code, cc.course_name,
c.special_fees FROM
student_info _<employee id> s, course_fees _<employee id> c,
course_info _<employee id> cc, student_courses _<employee id> sc
WHERE sc.student_id=s.student_id AND cc.course_code=c.course_code
AND sc.course_code=cc.course_code AND c.special_fees < 150;
```

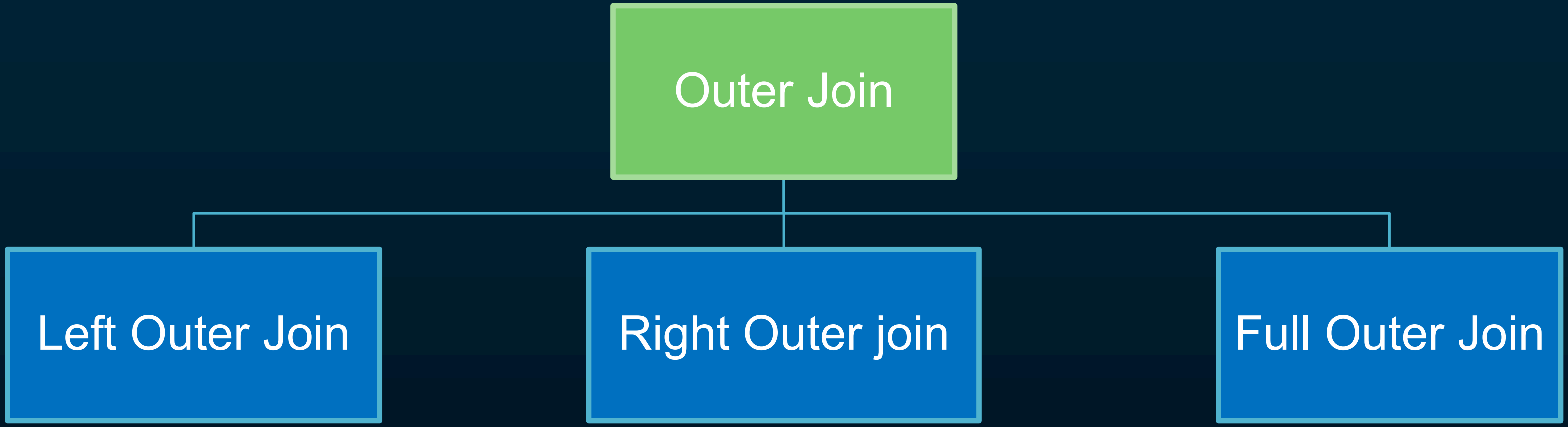
Outer Join

Outer Join

Outer-join

An ***outer join*** returns all rows from one table and only those rows from a secondary table where the join condition is met.

Types of Outer Join:



Left Outer Join

Left Outer Join

- In this join all rows from *table* specified on the left side of the join statement will appear.
- When no matching data is found from the table on the right side of the join, nulls are placed into fields.

Syntax:

```
SELECT column_name(s)
FROM table_1 LEFT OUTER JOIN  table_2
ON table_1.column_name=table_2.column_name
```

$$\underbrace{\text{table\_1.column\_name=table\_2.column\_name}}_{\text{Join statement}}$$

All the rows of table_1 will be fetched even if a matching row is not found in table_2

Left Outer Join Example

Let us consider an example,

- We have a table **PersonsInfo** which stores information like **Persons Id** ,**Last name , First name , Address** and **City.**
- We have another table **OrdersInfo** which stores information like **OrderId**, **Order number** and **Persons Id**.

```
SELECT PersonsInfo.LastName,PersonsInfo.FirstName,
OrdersInfo.OrderNo FROM PersonsInfo LEFT OUTER JOIN
OrdersInfo ON  PersonsInfo.P_Id= OrdersInfo.P_Id
```

Left Outer Join Example

PersonsInfo:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

OrdersInfo

P_Id	O_Id	OrderNo
3	1	77895
3	2	44678
1	3	22456
1	4	24562
15	5	34764

All the rows from **PersonsInfo** will be selected and matching rows from **OrdersInfo** will be selected.

Output:

LastName	FirstName	OrderNo
Pettersen	Kari	77895
Pettersen	Kari	44678
Hansen	Ola	22456
Hansen	Ola	24562
Svendson	Tove	<NULL>

Since this row does not has matching value in **OrdersInfo** table <null> value is added

Lend a Hand – Left Outer Join

Pre-requisite:

Create a student record (***student_info_*** <employee id>)

Create a course (***courses_info_*** <employee id>)

Enroll a course for that student (***student_courses_*** <employee id>)

NOTE:

No Course fee (Course_Fees_ <employee id>) needs to be added for the course.

Lend a Hand – Left Outer Join

Problem 1 :

Write a query to fetches the student id and base fees from

student_courses_<employee_id> , course_fees_<employee_id> table.

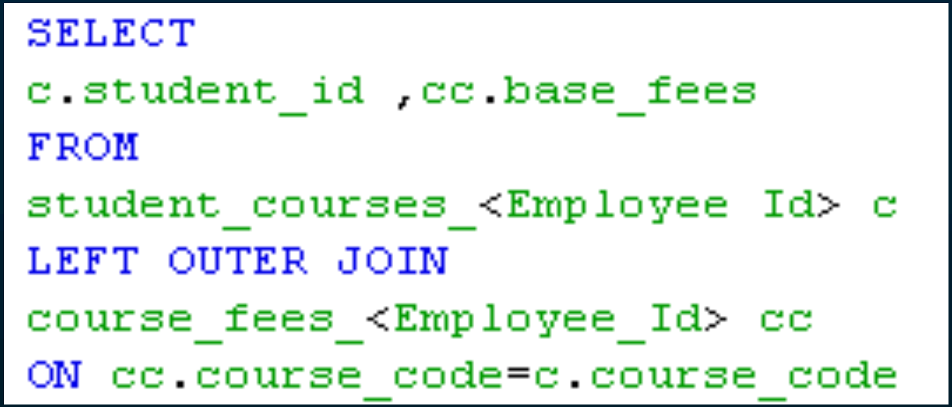
Note even if a course does not have fees, the record needs to be fetched.

Expected output: The student, course record which was added in the

system with no fees (in Pre-requisite) needs to be fetched.

Lend a Hand Solution– Left Outer Join

Solution # 1:



```
SELECT
c.student_id ,cc.base_fees
FROM
student_courses_<Employee Id> c
LEFT OUTER JOIN
course_fees_<Employee_Id> cc
ON cc.course_code=c.course_code
```

Right Outer Join

- In this join all rows from *table* specified on the right side of the join statement will appear.
- When no matching data is found from the table on the left side of the join, nulls are placed into fields.

Right Outer Join

Syntax:

```
SELECT column_name(s)
FROM table_1 RIGHT OUTER JOIN table_2
ON table_1.column_name=table_2.column_name
```

$\underbrace{\texttt{table\_1.column\_name=table\_2.column\_name}}_{\text{Join statement}}$

All the rows of table_2 will be fetched even if a matching row is not found in table_1

Right Outer Join Example

Let us consider an example:

- We have a table **PersonsInfo** which stores information like **Persons Id ,Last name , First name , Address** and **City.**
- We have another table **OrdersInfo** which stores information like **OrderId**, **Order number** and **Persons Id**.

Lets see how the below left outer join works?

```
SELECT PersonsInfo.LastName, PersonsInfo.FirstName,
OrdersInfo.OrderNo
FROM PersonsInfo RIGHT OUTER JOIN OrdersInfo
WHERE PersonsInfo.P_Id=OrdersInfo.P_Id
```

Right Outer Join Example

PersonsInfo:

P_Id	LastName	FirstName	Address	City
1	Hansen	Ola	Timoteivn 10	Sandnes
2	Svendson	Tove	Borgvn 23	Sandnes
3	Pettersen	Kari	Storgt 20	Stavanger

OrdersInfo:

P_Id	O_Id	OrderNo
3	1	77895
3	2	44678
1	3	22456
1	4	24562
15	5	34764

All the rows from **OrdersInfo** will be selected and matching rows from **PersonsInfo** will be selected.

Output:

LastName	FirstName	OrderNo
Pettersen	Kari	77895
Pettersen	Kari	44678
Hansen	Ola	22456
Hansen	Ola	24562
<NULL>	**<NULL>**	**34764**

Since this row does not has matching value in **PersonsInfo** table <null> value are added

Lend a Hand – Right Outer Join

Problem 1 :

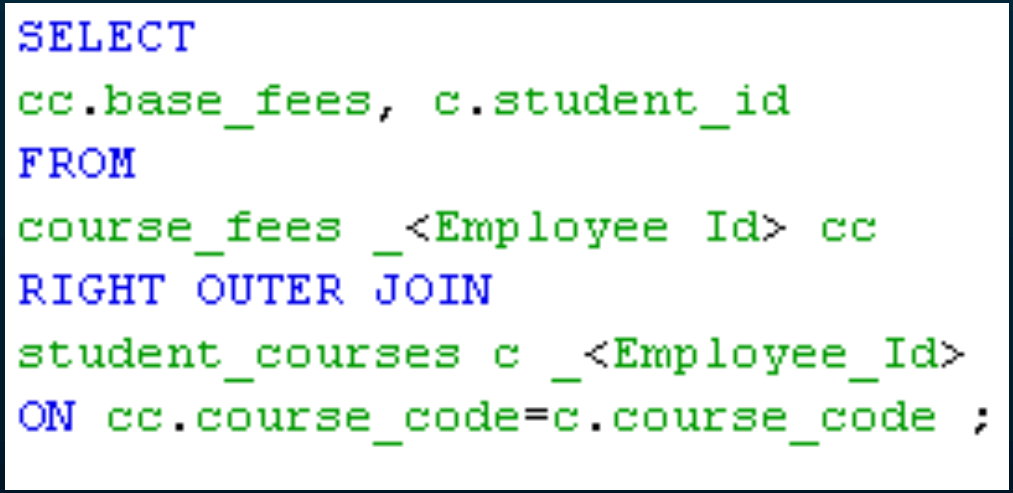
Write a query to fetches the student id and base fees from student_courses_<Employee Id> , course_fees_<Employee Id> table. Note even if a course does not have fees, the record needs to be fetched.

Expected output:

The student, course record which was added in the system with no fees (in Pre-requisite) needs to be fetched.

Lend a Hand Solution – Right Outer Join

Solution # 1:



```
SELECT
cc.base_fees, c.student_id
FROM
course_fees _<Employee Id> cc
RIGHT OUTER JOIN
student_courses c _<Employee_Id>
ON cc.course_code=c.course_code ;
```

Full Outer Join

- The ***Full Outer Join*** keyword returns all the rows from
  - table left to the join condition and
  - table right to the join condition
- From the tables **PersonsInfo** and **OrdersInfo** if we perform Full Outer Join
  - all the rows from **PersonsInfo** and **OrdersInfo** will be fetched
  - In addition, rows in "**PersonsInfo** " that do not have matches in "**OrdersInfo**" and rows in "**OrdersInfo**" that do not have matches in "**PersonsInfo** ", will also be selected extended with nulls.

LastName	FirstName	OrderNo
Pettersen	Kari	77895
Pettersen	Kari	44678
Hansen	Ola	22456
Hansen	Ola	24562
Svendson	Tove	**<NULL>**
<NULL>	**<NULL>**	**34764**

Self Join

self-join

- Self join is used in situations in which one row of a table is compared to another row of the same table.
- The table, on which the self join will be used, appears twice in the ***From*** clause
- We use table aliases to qualify column names in the join condition.

Self Join Example

Example:

We have table by name **Employee** which stores information like **employee_Id** , **employee_name** ,**salary** and **manager_id.**

NOTE: Manager_id is nothing but again a employee id in the same employee table.

Self Join Example

To find the manager of each employee let us perform self join:

```
SELECT w.ename || ' works for '|| ' ' || m.ename FROM EmployeeInformation w,
EmployeeInformation m
WHERE w.manager_id = m.employee_id
```

EMPNO	ENAME	SALARY	MANAGERID
1	Tom	14000	2
2	John	20000	
4	Joe	18000	1
5	Jane	7000	4
6	James	9000	5

Output:

```
Tom works for John
Joe works for Tom
Jodd works for James
Jane works for Joe
James works for Jane
```

ANSI/ISO SQL :1999- Compliant Joins

- These joins are different from traditional Oracle join syntax
- Join types are specified explicitly in from clause
- Some of the ANSI SQL: 1999-compliant joins are:
  - CROSS JOIN ……………. [WHERE (condition)]
  - [INNER] JOIN ……………… [ON(join condition)]
  - NATURAL JOIN ……………….[WHERE (condition)]

CROSS Join

- No join condition is specified
- Cross product of two tables, will contain duplicate columns in result
- Also called as Cartesian product

Example:

```
select LastName, FirstName, OrderNo from PersonsInfo
cross join
OrdersInfo where OrderNo = 77895;
```

This is equivalent to the following:

```
select LastName, FirstName, OrderNo from PersonsInfo ,OrdersInfo
where OrderNo =77895;
```

NATURAL join

- Selects rows from the tables which have equal values in all matched columns.
- If the columns have same names but different datatype , an error is returned.
- If select * syntax is used common columns will appear only once in the result.

NATURAL join

Example:

```
select LastName, FirstName, OrderNo from PersonsInfo
natural join OrdersInfo where P_Id =3;
```

This is equivalent to the following:

```
select LastName, FirstName, OrderNo from PersonsInfo, OrdersInfo
where
PersonsInfo.P_Id = OrdersInfo.P_Id and PersonsInfo.P_Id = 3;
```

Test your Understanding

1. What are the advantages of Joins?
2. How to retrieve only the matching rows from two tables?
3. How Outer Join is different from Inner Join?
4. When to use Self Join?

You have successfully completed
Displaying Data from Multiple Table.