Republic of Iraq
The Ministry of Higher Education
& Scientific Research
Al-Ma'arif University College
Department of Computer Science

Lecturer Name: Shihab Hamad Khaleefah
Academic Status: BhD. In Computer Science
Qualification: - Lecturer
Course Material: Database Management System
Grade Level: Second Stage, Lec. 10-12

The (RDD) & Join in (DMS)

1. Database Join

1.1 Database Join Important Terms

1.1.1 Foreign Key

1.1.2 Aliases Names

1.2 Database Join Types

1.2.1 (INNER) JOIN

1.2.2 (OUTER) JOIN

1.2.2.1 FULL (OUTER) JOIN

1.2.2.2 LEFT (OUTER) JOIN

1.2.2.3 RIGHT (OUTER) JOIN

1.2.3 The CARTESIAN JOIN or CROSS JOIN

1.3 A Full Example that Explain all Types of Join

2. Summary

1. SQL - Foreign Key

a. Define the Foreign Key

A foreign key is a key used to link two tables together. This is sometimes called a referencing key.

Foreign Key is a column or a combination of columns whose values match a Primary Key in a different table.

The relationship between 2 tables matches the Primary Key in one of the tables with a Foreign Key in the second table.

Foreign key (**column**) references table name that contains the primary key (**primary key column in main table**)

Simple example:

```
use auc_cs_dep;
create table info(
id int not null primary key identity (1,1),          //to set the began and increase value.
name varchar (255) not null,
age int not null
);
create table more (
mid int not null primary key,
email varchar (200) not null,
id int not null,
foreign key (id) references info(id)

)
```

Example:

Consider the structure of the two tables as follows:

CUSTOMERS table:

```
CREATE TABLE CUSTOMERS(
       ID   INT              NOT NULL,
       NAME VARCHAR (20)     NOT NULL,
       AGE  INT              NOT NULL,
       ADDRESS  CHAR (25) ,
       SALARY   DECIMAL (18, 2),
       PRIMARY KEY (ID)
);
```

ORDERS table:

```
CREATE TABLE ORDERS (
       ID          INT        NOT NULL,
       DATE        DATETIME,
       CUSTOMER_ID INT references CUSTOMERS(ID),
       AMOUNT     double,
       PRIMARY KEY (ID)
);
```

b. Add the Foreign Key:

If ORDERS table has already been created, and the foreign key has not yet been set, use the syntax for specifying a foreign key by altering a table.

```
ALTER TABLE ORDERS
   ADD FOREIGN KEY (Customer_ID) REFERENCES CUSTOMERS (ID);
```

c. DROP the FOREIGN KEY Constraint:

To drop a FOREIGN KEY constraint, use the following SQL:

```
ALTER TABLE ORDERS
   DROP FOREIGN KEY;
```

2. SQL - Creating Schema Objects

A *schema* is a collection of database objects (as far as this hour is concerned—tables) associated with one particular database username. This username is called the *schema owner*, or the owner of the related group of objects. You may have one or multiple schemas in a database.

Simple example:

```
Order Table
JOIN OrderItem I ON O.Id = I.OrderId
```

- The O, I are table aliases.

// Next lecture I will explain in depth details the Managing of Database Objects in SQL (Creating Schema Objects).

3. SQL – JOIN:

The SQL **Joins** clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each.

Consider the following two tables,

(a) CUSTOMERS table is as follows:

```
+----+----------+-----+-----------+----------+
| ID | NAME     | AGE | ADDRESS   | SALARY   |
+----+----------+-----+-----------+----------+
|  1 | Ramesh   |  32 | Ahmedabad |  2000.00 |
|  2 | Khilan   |  25 | Delhi     |  1500.00 |
|  3 | kaushik  |  23 | Kota      |  2000.00 |
|  4 | Chaitali |  25 | Mumbai    |  6500.00 |
|  5 | Hardik   |  27 | Bhopal    |  8500.00 |
|  6 | Komal    |  22 | MP        |  4500.00 |
|  7 | Muffy    |  24 | Indore    | 10000.00 |
+----+----------+-----+-----------+----------+
```

(b) Another table is ORDERS as follows:

```
+-----+---------------------+-------------+--------+
|OID  | DATE                | CUSTOMER_ID | AMOUNT |
+-----+---------------------+-------------+--------+
| 102 | 2009-10-08 00:00:00 |           3 |   3000 |
| 100 | 2009-10-08 00:00:00 |           3 |   1500 |
| 101 | 2009-11-20 00:00:00 |           2 |   1560 |
| 103 | 2008-05-20 00:00:00 |           4 |   2060 |
+-----+---------------------+------------+--------+
```

Now, let us join these two tables in our SELECT statement as follows:

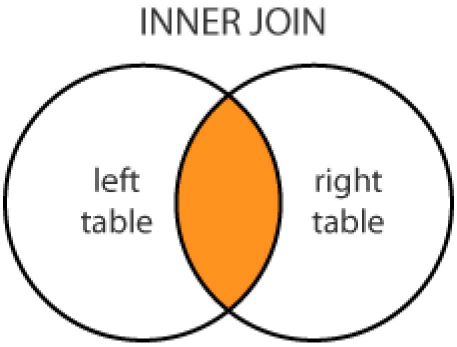
```
SQL> SELECT ID, NAME, AGE, AMOUNT
        FROM CUSTOMERS, ORDERS
        WHERE  CUSTOMERS.ID = ORDERS.CUSTOMER_ID;
```

This would produce the following result:

```
+----+----------+-----+--------+
| ID | NAME     | AGE | AMOUNT |
+----+----------+-----+--------+
|  3 | kaushik  |  23 |   3000 |
|  3 | kaushik  |  23 |   1500 |
|  2 | Khilan   |  25 |   1560 |
|  4 | Chaitali |  25 |   2060 |
+----+----------+-----+--------+
```

* Important Notes:

- A SQL JOIN combines records from two tables.
- A JOIN locates related column values in the two tables.
- A query can contain zero, one, or multiple JOIN operations.
- INNER JOIN is the same as JOIN; the keyword INNER is optional.



4. The different types of JOINs

i. (INNER) JOIN: Select records that have matching values in both tables.

Which is the default type, it determines that if the record from two different tables are compatible, but they agree ON clause that binds them, then you should include it in the data set, otherwise neglected, for example:

```
SELECT   s.stor_id, d.discounttype
FROM     stores s JOIN discounts d
ON       s.stor_id = d.stor_id
```

Here are merging the two tables stores and discounts to determine which stores are providing a discount on its product, and display the type of reduction, this query can be written in another way are:

```
select s.store_id, d.discounttype
from stores s, discounts d
where s.store_id=d.store_id
```

ii. (OUTER) JOIN: It is divided into three main types ….

- FULL (OUTER) JOIN: Selects all records that match either left or right table records.

It specifies that all not matched (that do not meet the requirement ON) and matched records (which meet the condition) are selected. For the non-matched records will appear null value which, in this example, the appearance of null value means that the store doesn't not provide any discounts on its product. This is because store_id value in the record of the stores table do not match the value in the record of the store_id in discounts table, for example:

```
SELECT   s.stor_id, d.discounttype
FROM     stores s FULL OUTER JOIN discounts d
ON       s.stor_id = d.stor_id
```

- LEFT (OUTER) JOIN: Select records from the first (left-most) table with matching right table records.

retrieve all records that match the condition with all records of the selected table (left of the join word), for example:

```
SELECT    s.stor_id, d.discounttype
FROM      stores s LEFT OUTER JOIN discounts d
ON        s.stor_id = d.stor_id
```

// Null value will also appear in **discount type** field in the records that the value of **store_id** where not match with the requirement of joining.

- RIGHT (OUTER) JOIN: Select records from the second (right-most) table with matching left table records.

retrieve all records that match the condition with all records of the selected table (Right of the join word), for example:

```
SELECT    s.stor_id, d.discounttype
FROM      stores s RIGHT OUTER JOIN discounts d
ON        s.stor_id = d.stor_id
```

// Here null value appears in all the records of the discount type table that does not have the store_id value matching in stores table exactly in the store_id record, While in the left outer join example, it appears in the discount type records.

iii. **The CARTESIAN JOIN or CROSS JOIN** returns the Cartesian product of the sets of records from the two or more joined tables. Thus, it equates to an inner join where the join-condition always evaluates to True or where the join-condition is absent from the statement.

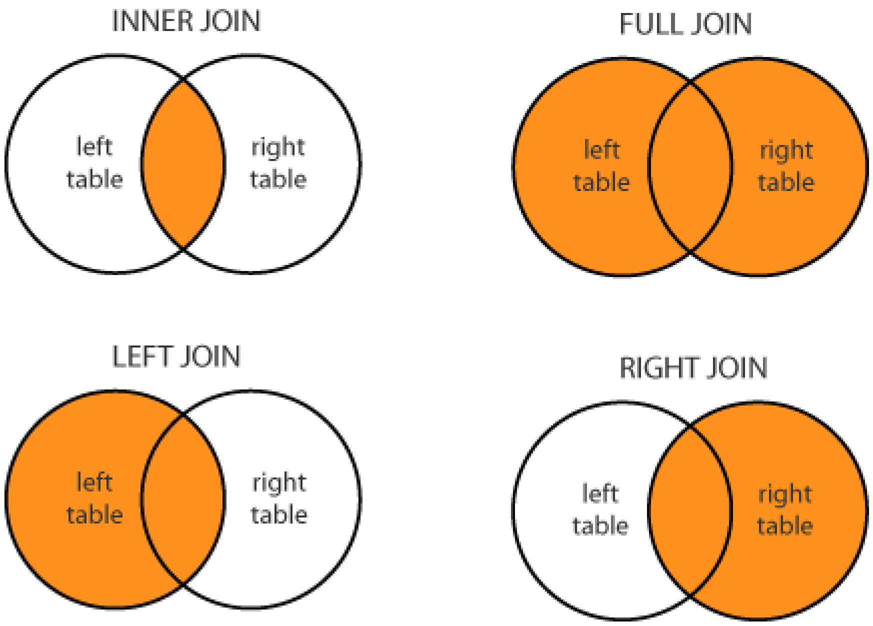
A selection quotient of both of the two tables in the case of non-select a paragraph where, in this case each record of the first table will join with each record in the second table, so the number of records resulting by applying cross join query = number of the first table records multiplied by the number of second table records (cartesian product).

// when we used where it will be like inner join.

```
SELECT   *  FROM    stores CROSS JOIN sales
```

Or

```
SELECT   * FROM    stores, sales
```



Important Note: All INNER and OUTER keywords are optional: it is the default as well as the most commonly used JOIN operation.

5. The SQL JOIN syntax

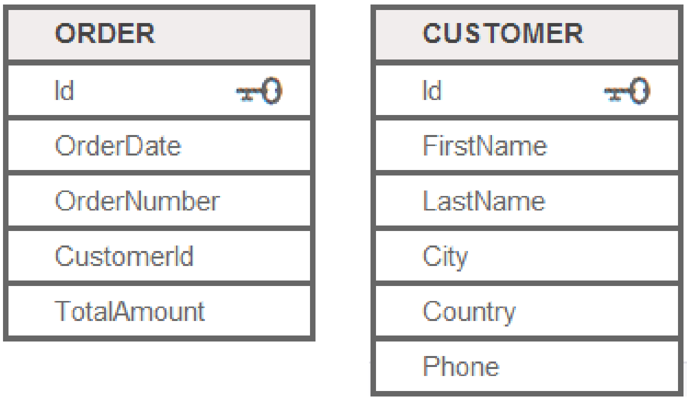
- The general syntax is:

```
SELECT column-names
  FROM table-name1 JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

- The general syntax with INNER is:

```
SELECT column-names
  FROM table-name1 INNER JOIN table-name2
    ON column-name1 = column-name2
 WHERE condition
```

SQL JOIN Examples



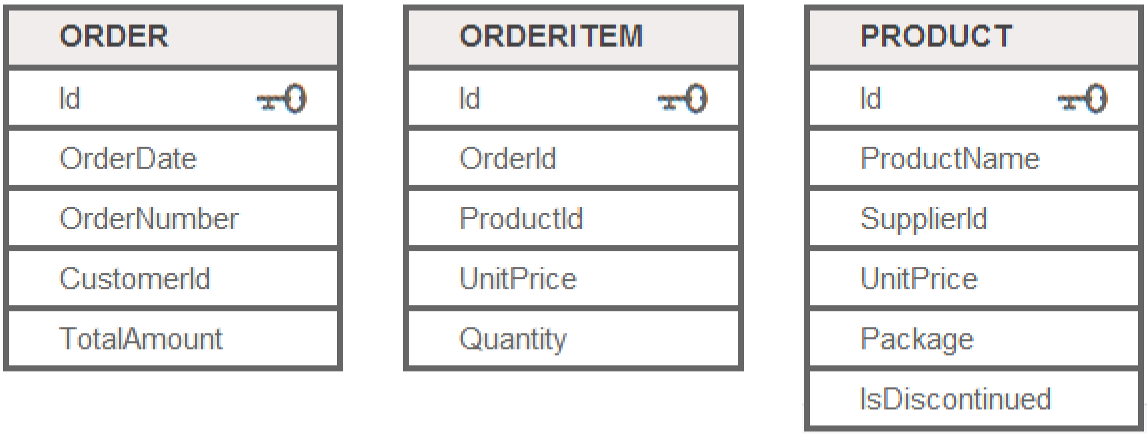
Problem: List all orders with customer information

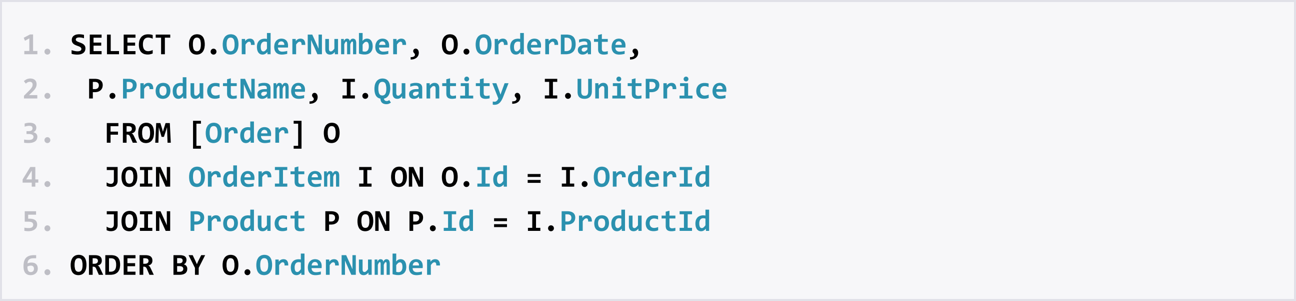
```
SELECT OrderNumber, TotalAmount, FirstName, LastName, City, Country
  FROM [Order] JOIN Customer
    ON [Order].CustomerId = Customer.Id
```

In this example using table aliases for [Order] and Customer might have been useful.

OrderNumber	TotalAmount	FirstName	LastName	City	Country
542378	440.00	Paul	Henriot	Reims	France
542379	1863.40	Karin	Josephs	Münster	Germany
542380	1813.00	Mario	Pontes	Rio de Janeiro	Brazil
542381	670.80	Mary	Saveley	Lyon	France
542382	3730.00	Pascale	Cartrain	Charleroi	Belgium
542383	1444.80	Mario	Pontes	Rio de Janeiro	Brazil
542384	625.20	Yang	Wang	Bern	Switzerland
...					

Problem: List all orders with product names, quantities, and prices



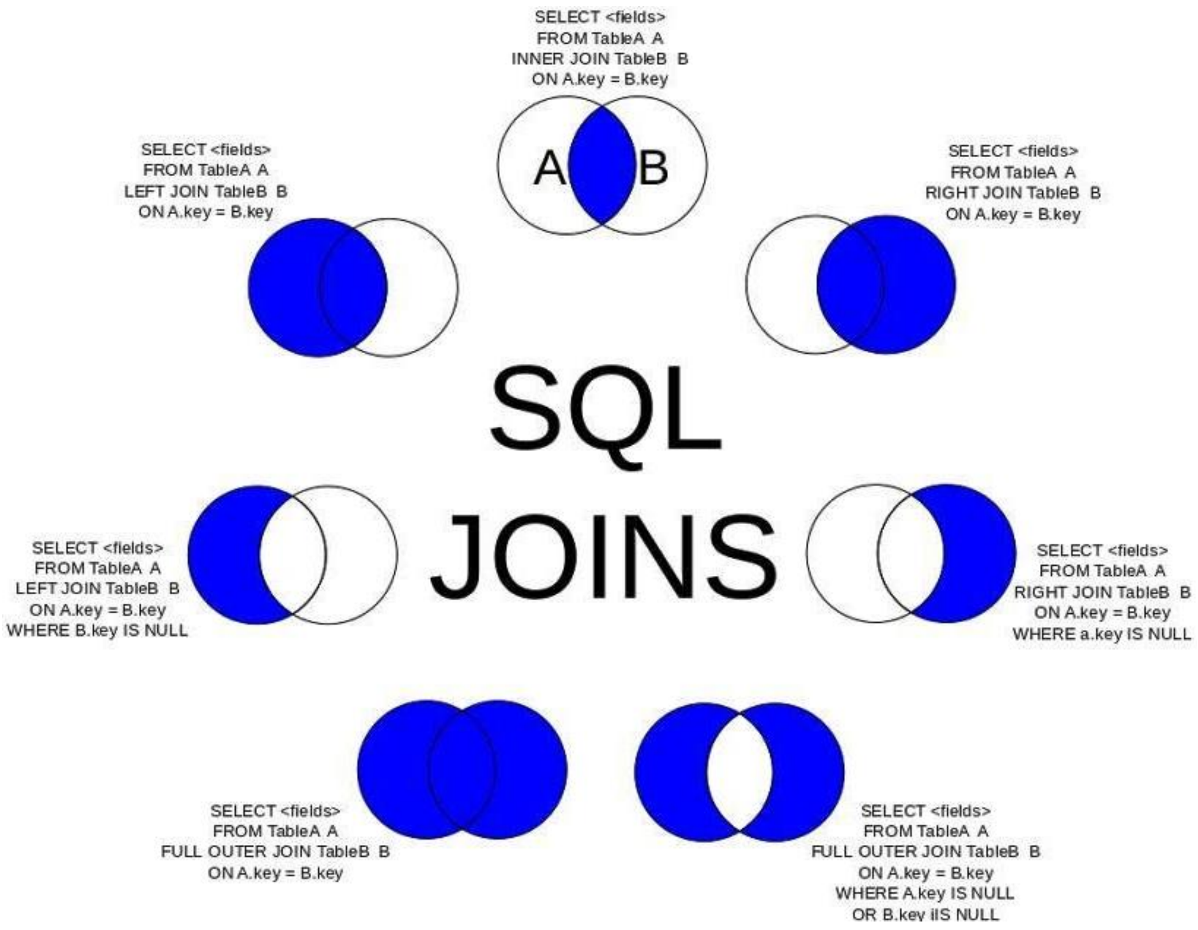


```
SELECT O.OrderNumber, O.OrderDate,
 P.ProductName, I.Quantity, I.UnitPrice
  FROM [Order] O
  JOIN OrderItem I ON O.Id = I.OrderId
  JOIN Product P ON P.Id = I.ProductId
ORDER BY O.OrderNumber
```

This query performs two JOIN operations with 3 tables.
The O, I, and P are table aliases. Date is a column alias.

OrderNumber	Date	ProductName	Quantity	UnitPrice
542378	7/4/2012 12:00:00 AM	Queso Cabrales	12	14.00
542378	7/4/2012 12:00:00 AM	Singaporean Hokkien Fried Mee	10	9.80
542378	7/4/2012 12:00:00 AM	Mozzarella di Giovanni	5	34.80
542379	7/5/2012 12:00:00 AM	Tofu	9	18.60
542379	7/5/2012 12:00:00 AM	Manjimup Dried Apples	40	42.40

542380	7/8/2012 12:00:00 AM	Jack's New England Clam Chowder	10	7.70
542380	7/8/2012 12:00:00 AM	Manjimup Dried Apples	35	42.40
542380	7/8/2012 12:00:00 AM	Louisiana Fiery Hot Pepper Sauce	15	16.80
542381	7/8/2012 12:00:00 AM	Gustaf's Knäckebröd	6	16.80
542381	7/8/2012 12:00:00 AM	Ravioli Angelo	15	15.60
542381	7/8/2012 12:00:00 AM	Louisiana Fiery Hot Pepper Sauce	20	16.80
542382	7/9/2012 12:00:00 AM	Sir Rodney's Marmalade	40	64.80
542382	7/9/2012 12:00:00 AM	Geitost	25	2.00



```
SELECT <fields>
FROM TableA A
INNER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
LEFT JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
```

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

```
SELECT <fields>
FROM TableA A
RIGHT JOIN TableB B
ON A.key = B.key
WHERE a.key IS NULL
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
```

```
SELECT <fields>
FROM TableA A
FULL OUTER JOIN TableB B
ON A.key = B.key
WHERE A.key IS NULL
OR B.kev iIS NULL
```

Summary

In this lecture,

After the students learned how to deal with many of functions and operators in data definition language (DDL), data manipulation language (DML) and data query language (DQL) operators in structured query language (SQL) with entire examples that illustrate the work of its functions and operators. Now, they will learn how to deal with all types of Joining.

They will be able to learn how to deal with the important terms such as: Foreign Key, Aliases Names. Then they will be able to deal with the Database Join Types which are: (Inner) Join, (Outer) Join [Full (Outer) Join, Left (Outer) Join & Right (Outer) Join], Finally, The Cartesian Join or Cross Join. After that a full Example that Explain all Types of Join.