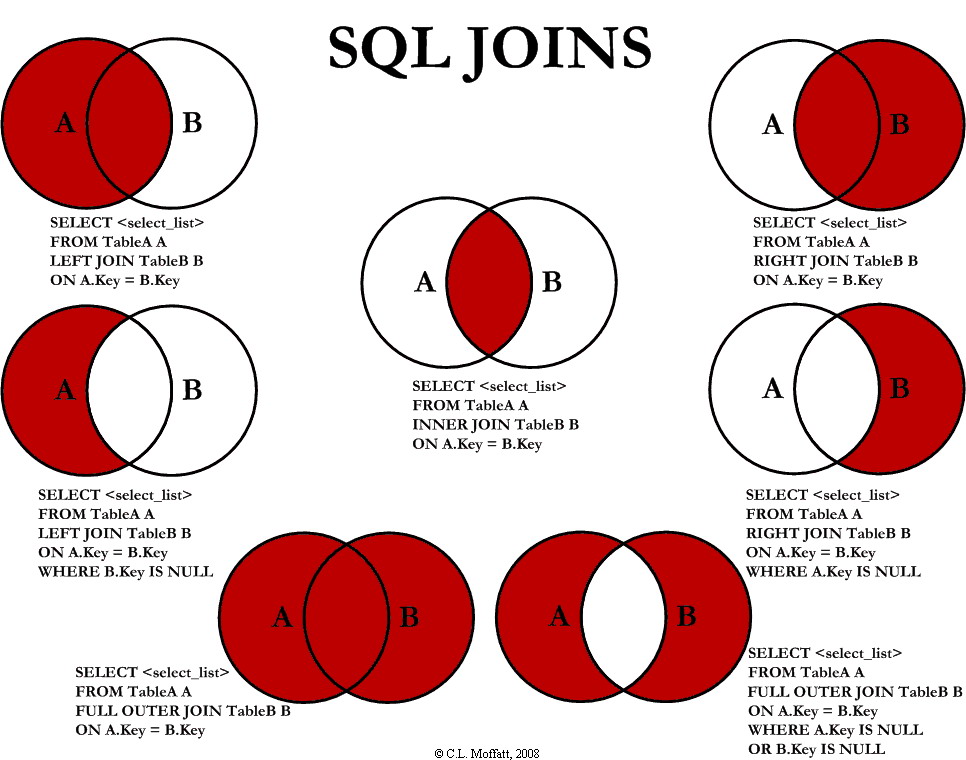
Tips and Hints: Using SQL Joins



Supporting Resources:

[**MSDN - SQL JOINS**](https://docs.microsoft.com/en-us/sql/relational-databases/performance/joins?view=sql-server-2017) A query result can include data from multiple tables or table-valued objects. To combine data from multiple table-valued objects, you use the JOIN operation from SQL. SQL Server performs sort, intersect, union, and difference operations using in-memory sorting and hash join technology. Using this type of query plan, SQL Server supports vertical table partitioning, sometimes called columnar storage.

Joins in SQL Server

|  |  |
| --- | --- |
| [Graphical user interface, application  Description automatically generated](https://www.youtube.com/watch?v=hfFznlHjrEQ&t=5s) | TU SQL Server Developer Group  [Youtube Video: SQL Joins](https://www.youtube.com/watch?v=hfFznlHjrEQ&t=5s) |

An **SQL join** clause - corresponding to a **join** operation in relational algebra - combines columns from one or more tables in a relational database. It creates a set that can be saved as a table or used as it is. A **JOIN** is a **means** for combining columns from one (self-**join**) or more tables by using values common to each.

Generally, SQL practitioners and Developers do not appreciate the power of joins. It is how tables can be related to each other. A powerful tool for mapping relationships across many tables.

**The best ICA sample solutions are best demonstrated using data from related tables.**

If you know the different types of JOIN in SQL Server and how they can be best to obtain data from many tables based on some logical conditions.

The Different Types of Joins in SQL Server

There are only 4 kinds:

1. **Inner join**: The most common type. An output row is produced for every pair of input rows that match on the join conditions.
2. **Left outer join**: The same as an inner join, except that if there is any row for which no matching row in the table on the right can be found, a row is output containing the values from the table on the left, with NULL for each value in the table on the right. This means that every row from the table on the left will appear at least once in the output.
3. **Right outer join**: The same as a left outer join, except with the roles of the tables reversed.
4. **Full outer join**: A combination of left and right outer joins. Every row from both tables will appear in the output at least once.

Joins by Examples

Let's suppose we have two tables Employee and Department whose description is given below:-

CREATE TABLE [dbo]. [Employee](

[Empid] [Int] IDENTITY (1, 1) NOT NULL Primary key,

[EmpNumber] [nvarchar](50) NOT NULL,

[EmpFirstName] [nvarchar](150) NOT NULL,

[EmpLastName] [nvarchar](150) NULL,

[EmpEmail] [nvarchar](150) NULL,

[Managerid] [int] NULL,

[Departmentid] [INT]

)

CREATE TABLE [dbo].[Department](

[Departmenttid] [int] IDENTITY (1, 1) NOT NULL primary key,

[DepartmentName] [nvarchar](255) NOT NULL

)

After the creation of the tables we need to insert the data into these tables. To insert the data the following queries are used:-

insert into Employee

(EmpNumber,EmpFirstName,EmpLastName,EmpEmail,Managerid,Departmentid)

values('A001','Mansh','Nawaz','Mansh@abc.com',2,2)

insert into Employee

(EmpNumber,EmpFirstName,EmpLastName,EmpEmail,Managerid,Departmentid)

values('A002','Alex','Groves','Alex@abc.com',1,1)

insert into Employee (EmpNumber,EmpFirstName,EmpLastName,EmpEmail,Managerid,Departmentid)

values('A003','Erika','Downs','Erika@abc.com',1,2)

insert into Employee (EmpNumber,EmpFirstName,EmpLastName,EmpEmail,Managerid,Departmentid)

values('A004','Tyrone','Davis','Tyrone@abc.com',1,NULL)

insert into Department(DepartmentName)

values('Accounts')

insert into Department(DepartmentName)

values('Admin')

insert into Department(DepartmentName)

values('HR')

insert into Department(DepartmentName)

values('Technology')

Inner Join

This type of join is also known as the Equi join. This join returns all the rows from both tables where there is a match. This type of join can be used in the situation where we need to select only those rows which have values common in the columns which are specified in the ON clause.

Now, if we want to get employee id, employee first name, employee's last name and their department name for those entries employee which belongs to at least one department, then we can use the inner join.

Query for Inner Join

SELECT Emp.Empid, Emp.EmpFirstName, Emp.EmpLastName, Dept.DepartmentName

FROM Employee Emp

INNER JOIN Department dept

ON Emp.Departmentid=Dept.Departmenttid

Result

Empid EmpFirstName EmpLastName DepartmentName

1 Mansh Nawaz Admin

2 Alex Groves Accounts

3 Erika Downs Admin

Explanation

In this query, we used the inner join based on the column "Departmentid" which is common in both the tables "Employee" and "Department". This query will give all the rows from both the tables which have common values in the column "Departmentid". Erika Downs and Mansh Nawaz has the value "2" in the Departmentid column of the table Employee. In the Department table, the Department "Admin" has the value "2" in the Departmentid column. Therefore the above query returns two rows for the department "Admin", one for Erika Downs and another for Mansh Nawaz.

Self Join

Sometime we need to join a table to itself. This type of join is called Self join. It is one of the type of inner join where both the columns belong to the same table. In this Join, we need to open two copies of a same table in the memory. Since the table name is the same for both instances, we use the table aliases to make identical copies of the same table to be open in different memory locations. For example if we need to get the employee name and their manager name we need to use the self join, since the **managerid** for an employee is also stored in the same table as the employee.

Query for the Self Join

SELECT Emp1.Empid,

Emp1.EmpFirstName+' '+Emp1.EmpLastName as EmployeeName,

Emp2.EmpFirstName+' '+Emp2.EmpLastName as ManagerName

FROM Employee Emp1

INNER JOIN Employee Emp2

ON Emp1.Managerid=Emp2.Empid

Result

Empid EmployeeName ManagerName

1 Mansh Nawaz Alex Groves

2 Alex Groves Mansh Nawaz

3 Erika Downs Mansh Nawaz

4 Tyrone Davis Mansh Nawaz

Explanation

Since the employee and the manager information is contained in the same table (Employee, since both are employees), we have to use the Self Join. In the self join query, we make two copies of the table Employee by using the aliases Emp1 and Emp2 and then use Inner join between them by using the managerid column of the Emp1 and Empid column of the table Emp2.In this example, we use managerid and empid columns of the Employee table since the employee id of the manager of an employee is stored in the managerid of the Employee table.

Outer Join

This type of join is needed when we need to select all the rows from the table on the left (or right or both) regardless of whether the other table has common values or not and it usually enter null values for the data which is missing.

The Outer join can be of three types

1. **Left Outer Join**
2. **Right Outer Join**
3. **Full Outer Join**

Left Outer Join

If we want to get employee id, employee first name, employes last name and their department name for all the employees regardless of whether they belong to any department or not,then we can use the left outer join. In this case we keep the Employee table on the left side of the join clause. It will insert NULL values for the data which is missing in the right table.

Query for Left Outer Join

SELECT Emp.Empid,

Emp.EmpFirstName,

Emp.EmpLastName,

Dept.DepartmentName

FROM Employee Emp

LEFT OUTER JOIN Department dept

ON Emp.Departmentid=Dept.Departmenttid

Result

Empid EmpFirstName EmpLastName DepartmentName

1 Mansh Nawaz Admin

2 Alex Groves Accounts

3 Erika Downs Admin

4 Tyrone Davis NULL

Explanation

Since we have use the Left Outer Join, this query will give the information (Employee id, Employee first name, Employee last name and their department name) for all the employee from the Employee table and it insert NULL value in the DepartmentName column where the employee does not belong to any department. In the table Employee, since Mansh Nawaz, Alex Groves and Erika Downs have values in their Departmentid column, therefore the above query will display their Department name under the heading DepartmentName.But since Tyrone Davis doesn't belongs to any department and has null value in the column Departmentid therefore the above query will Display the NULL value under the column heading DepartmentName.

Right Outer Join

If we want to get all the departments name and employee id, employee first name, and employees last name of all the employees belonging to the department regardless of whether a department have employees or not, then we can use the right outer join. In this case we keep the Department table on the right side of the join clause. It will insert NULL values for the data which is missing in the left table (Employee).

Query for Right Outer Join

SELECT Dept.DepartmentName,

Emp.Empid, Emp.EmpFirstName,

Emp.EmpLastName

FROM Employee Emp

RIGHT OUTER JOIN Department dept

ON Emp.Departmentid=Dept.Departmentid

Result

DepartmentName Empid EmpFirstName EmpLastName

Accounts 2 Alex Groves

Admin 1 Mansh Nawaz

Admin 3 Erika Downs

HR NULL NULL NULL

Technology NULL NULL NULL

Explanation

Since we have use the Right Outer Join, this query will join the two tables Employee and Department on the basis of the values contains in the column Departmenttid. It will give the department name from the Department table and the Employee id, Employee first name, and Employee last name of all the employees that belong to that department. If any department does not contain any employee then it insert NULL value in the columns coming from the Employee table. Since no employee is connected to the departments HR and Technology, this query will display NULL values under the columns Empid, EmpFirstName and EmpLastName for the Departments HR and Technology. Since the department Admin and Accounts contains the employees therefore the columns Empid, EmpFirstName and EmpLastName contains the information, employee id, employee first name and employee last name respectively.

Full Outer Join

If we want to get all the departments name and the employee id, employee first name, employes last name of all the employees regardless of whether a department have employees or not, or whether a employee belong to a department or not, then we can use the full outer join. It will insert null values for the data which is missing in both the tables.

Query for Full Outer Join

SELECT Emp.Empid,

Emp.EmpFirstName,

Emp.EmpLastName,

Dept.DepartmentName

FROM Employee Emp

FULL OUTER JOIN Department dept

ON Emp.Departmentid=Dept.Departmenttid

Result

Empid EmpFirstName EmpFirstName DepartmentName

1 Mansh Nawaz Admin

2 Alex Groves Accounts

3 Erika Downs Admin

4 Tyrone Davis NULL

NULL NULL NULL HR

NULL NULL NULL Technology

Explanation

Since we have used the Full Outer Join, this query will give the name of all the departments from the Department table and the Employee id, Employee first name, Employee last name of all the employees from the Employee table. If any department does not contain any employee, then it insert NULL value in the columns Empid, EmpFirstName, EmpLastName columns and if any employee doesn't belong to any department then it insert NULL value in the column DepartmentName. Here since Tyrone Davis doesn't belong to any department, the result displays NULL value under the column DepartmentName. Since the departments HR and Accounts don't contain any employees, the result of the above query displays NULL values under the columns Empid, EmpFirstName and EmpLastName for the departments HR and Technology..

Cross Join

This join combines all the rows from the left table with every row from the right table. This type of join is needed when we need to select all the possible combinations of rows and columns from both the tables. This type of join is generally not preferred as it takes lot of time and gives a huge result that is not often useful.

Query for the Cross Join

SELECT Emp.Empid,

Emp.EmpFirstName,

Emp.EmpLastName,

Dept.DepartmentName

FROM Employee Emp

CROSS JOIN Department dept

Results

Empid EmpFirstName EmpLastName DepartmentName

1 Mansh Nawaz Accounts

2 Alex Groves Accounts

3 Erika Downs Accounts

4 Tyrone Davis Accounts

1 Mansh Nawaz Admin

2 Alex Groves Admin

3 Erika Downs Admin

4 Tyrone Davis Admin

1 Mansh Nawaz HR

2 Alex Groves HR

3 Erika Downs HR

4 Tyrone Davis HR

1 Mansh Nawaz Technology

2 Alex Groves Technology

3 Erika Downs Technology

4 Tyrone Davis Technology

Explanation

This Cross Join query will give combines all the rows from the Employee table with every row of the Department table. Since the Employee table contains 4 rows and the Department table contains 4 rows, therefore this result will returns 4\*4=16 rows. This query doesn't contain any ON clause.

Wrapping Up

The above discussion can be summarized as joins are used to select data from more than one table in a single query. The inner join is used to select only those rows that have common values in the column on which join is based. The Left Outer Join is used to select the rows from the left hand side table regardless of whether the table on the right hand side has common values or not. Similarly the Right Outer join is used to select rows from the table on the right hand side regardless of whether the table on the left hand side has common values or not. The Cross join is used to get rows from all the possible combinations of rows and columns from both the table. If should be used when it the only way left since it may run for a very long time and returns a huge result set which may not be useful.

Joining of the tables should be avoided if it is based on the columns that have very few unique values. To increase the JOIN performance it is better to limits the number of rows needed to be joined, by including a WHERE clause in your query. Join performance can also we increased if the columns used for joining the tables have their own indexes.