|  |
| --- |
| **Join** |
|  |

A join is a technique of creating a list of records from more one table, using all columns from all tables involved, or selecting only the desired columns from one or all of the tables involved. This means that a data join is essentially created in three steps:

1. Selecting the tables that will be involved in the join
2. Selecting a column that will create the link in each table
3. Writing a [SQL statement](http://www.functionx.com/sqlserver/Lesson25.htm) that will create the records

Before creating a join, it is also important to create a primary key for each table. The parent table would usually need only this primary key that would be used to "link" it to a child table. When creating the child table, remember to create a column that would serve as the link with the parent table. The name and the data type of this column are the same as the primary key of the parent table.

USE Master;

GO

-- ===================================================

-- Database: People

-- Author: FunctionX.com

-- Date Created: Tuesday, July 28, 2009

-- ===================================================

IF EXISTS(SELECT name FROM sys.databases WHERE name = N'People')

DROP DATABASE People;

GO

CREATE DATABASE People;

GO

USE People;

GO

IF OBJECT\_ID('Genders', 'U') IS NOT NULL

DROP TABLE Genders;

GO

-- ===================================================

-- Database: People

-- Table: Genders

-- Author: FunctionX.com

-- Date Created: Tuesday, July 28, 2009

-- Description: This table holds the list of genders

-- ===================================================

CREATE TABLE Genders

(

GenderID int identity(1, 1) not null,

Gender nchar(15),

CONSTRAINT PK\_Genders PRIMARY KEY(GenderID)

);

GO

INSERT INTO Genders(Gender)

VALUES

(N'Male'),

(N'Female'),

(N'Unknown');

GO

IF OBJECT\_ID('Persons', 'U') IS NOT NULL

DROP TABLE Persons;

GO

-- ===================================================

-- Database: People

-- Table: Persons

-- Author: FunctionX.com

-- Date Created: Tuesday, July 28, 2009

-- Description: This table holds a list of people and their genders

-- ===================================================

CREATE TABLE Persons

(

PersonID int identity(1, 1) not null,

FirstName nvarchar(20),

LastName nvarchar(20),

GenderID int,

CONSTRAINT PK\_Persons PRIMARY KEY(PersonID)

);

GO

INSERT INTO Persons(FirstName, LastName, GenderID)

VALUES

(N'John', N'Franks', 1),

(N'Peter', N'Sonnens', 1),

(N'Mary', N'Shamberg', 2),

(N'Chryssa', N'Lurie', 2),

(N'Hellah', N'Zanogh', 3),

(N'Arnie', N'Ephron', 3);

GO

INSERT INTO Persons(FirstName, LastName)

VALUES

(N'Helene', N'Campo'),

(N'Arnie', N'Ephron');

GO

INSERT INTO Persons(FirstName)

VALUES

(N'Robert');

GO

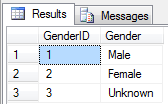
INSERT INTO Persons(LastName)

VALUES

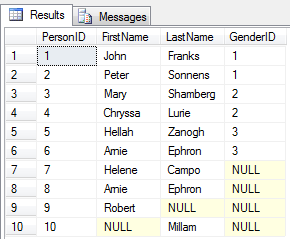
(N'Millam');

GO

SELECT \* FROM Genders



SELECT \* FROM Persons



|  |
| --- |
| **Cross Join** |
|  |

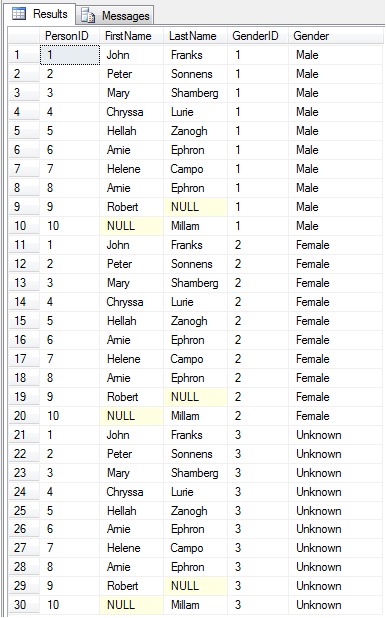
A cross join creates a list of all records from both tables as follows: the first record from the parent table is associated to each record from the child table, then the second record from the parent table is associated to each record from the child table, and so on. In this case also, there is no need of a common column between both tables. In other words, you will not use the **ON** clause.

SELECT P.PersonID, P.FirstName, P.LastName,

G.GenderID, G.Gender

FROM Persons P

CROSS JOIN Genders G



|  |
| --- |
| **Inner Join** |
|  |

Imagine you have two tables that can be linked through one's primary key and another's foreign key:

Notice that some records in the Persons table don't have an entry for the GenderID column and were marked with *NULL* by the database engine. When creating a query of records of the Persons table, if you want your list to include only records that have an entry, you can create it as inner join.

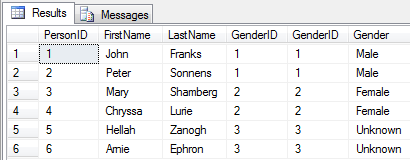
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

INNER JOIN Genders G

ON P.GenderID = G.GenderID



**JOIN and INNER JOIN are the same. 'INNER' is optional: the default JOIN is INNER JOIN.**

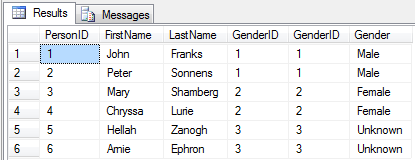
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

JOIN Genders G

ON P.GenderID = G.GenderID



|  |
| --- |
| **Outer Join** |
|  |

Instead of showing only records that have entries in the child table, you may want your query to include all records, including those that are null. To get this result, you would create an outer join. You have three options.

|  |
| --- |
| **Left Outer Join** |
|  |

A left outer join produces all records of the child table, also called the right table. The records of the child table that don't have an entry in the foreign key column are marked as *NULL*.

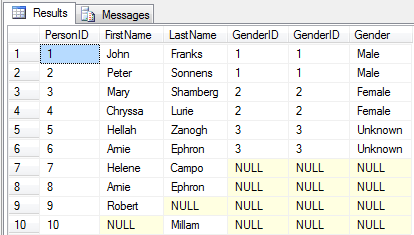
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

LEFT OUTER JOIN Genders G

ON P.GenderID = G.GenderID



**'OUTER' is optional in LEFT OUTER JOIN**

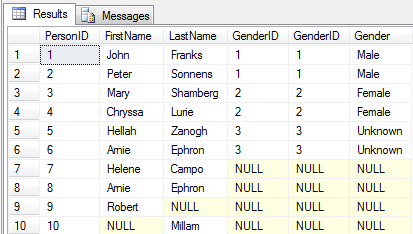
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

LEFT JOIN Genders G

ON P.GenderID = G.GenderID



|  |
| --- |
| **Right Outer Join** |
|  |

A right outer join considers all records from the parent table and finds a matching record in the child table. To do this, it starts with the first record of the parent table (in this case the Genders table) and shows each record of the child table (in this case the Persons table) that has a corresponding entry. This means that, in our example, a right outer join would first create a list of the Persons records that have a 1 (Female) value for the GenderID column. After the first record, the right outer join moves to the second record, and so on, each time listing the records of the child table that have a corresponding entry for the primary key of the parent table.

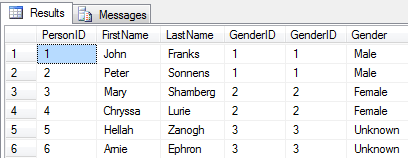
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

RIGHT OUTER JOIN Genders G

ON P.GenderID = G.GenderID



Notice that the query result starts with the first record of the parent table, also called the left table (in this case the Genders table), and lists the records of the child table, also called the right table (in this case the Persons table), that have the entry corresponding to that first record. Then it moves to the next GenderID value. Also, notice that there are no *NULL* records in the Gender.

**'OUTER' is optional in RIGHT OUTER JOIN**

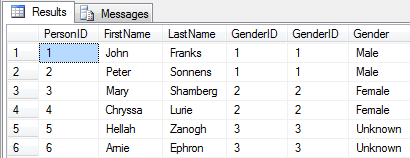
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

RIGHT JOIN Genders G

ON P.GenderID = G.GenderID



|  |
| --- |
| **Full Outer Join** |
|  |

A full outer join produces all records from both the parent and the child tables. If a record from one table doesn't have a corresponding value in the other table, the value of that record is marked as *NULL*.

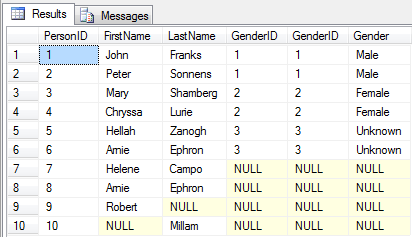
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

FULL OUTER JOIN Genders G

ON P.GenderID = G.GenderID



**'OUTER' is optional in FULL OUTER JOIN**

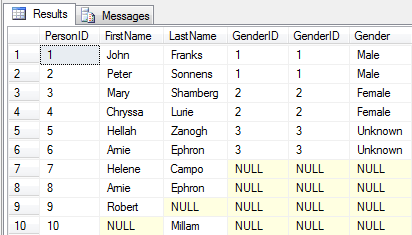
SELECT P.PersonID, P.FirstName, P.LastName, P.GenderID,

G.GenderID , G.Gender

FROM Persons P

FULL JOIN Genders G

ON P.GenderID = G.GenderID



Note: You can create a join that includes many tables.

<http://www.functionx.com/sqlserver/Lesson25.htm>