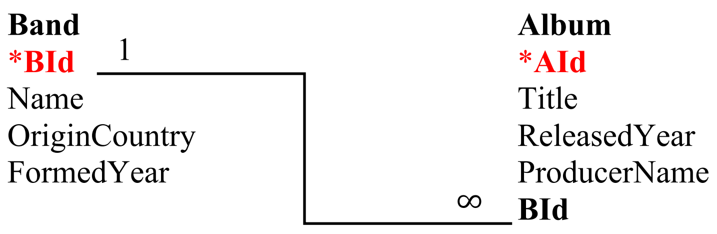
Create a table with a Foreign Key Constraint in SQL Server

Foreign Key constraint

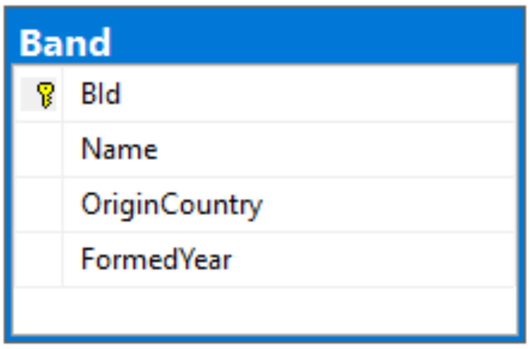
Our main objective is to present different ways in which a Foreign Key constraint can be created in a table.
The Foreign Key constraint describe the most important relationship that can be defined for two tables in a database: relationship 1:n (one-to-many). Relationship 1: n involves two tables: parent table (the part 1 of the relationship 1:n, in which is defined the Primary Key) and child table (the part n of the relationship 1:n, in which is defined the corresponding Foreign Key). In an equivalent manner, we can simply say that creating a Foreign Key constraint means creating the relationship 1:n between the two tables involved. Obviously, the table that contains the Primary Key should be created first and only after the table that contains the Foreign Key.
Primary `Key can be defined as the field / attribute from the table that has unique values for each record introduced in the table and it is not NULL.
Foreign key can be defined as the field / attribute from the corresponding table to which establish the relationship 1: n, having the *same type* as the Primary Key (int, float, varchar, …), the *same values* as the Primary Key (but, can appear more than once in the table in which is Foreign Key; e.g. the same parent can have multiple children) and *it is not NULL*.
Sometimes the name of the fields / attributes / columns that are set to be Primary Key and Foreign Key is the same, but there is no rule related to this aspect.

In what follows, we consider the relationship *Band – Album (1:n)* “A band records multiple albums and an album is recorded by a single band”, with the following description and structure:

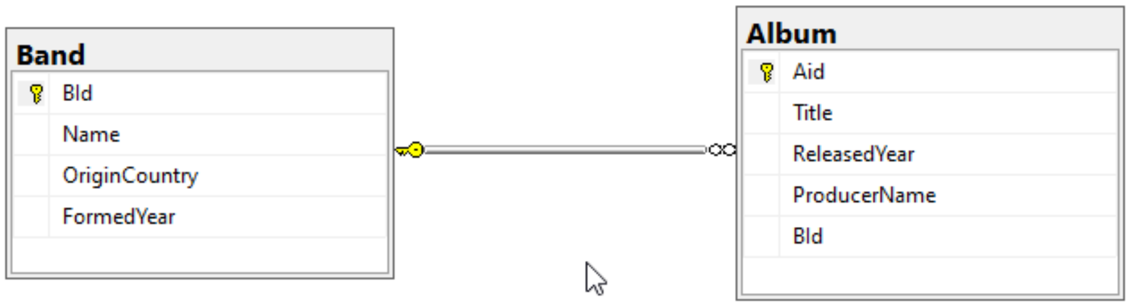


We used the notation * to mark the primary keys from the tables involved: *BId* for *Band* table and *AId* for *Album* table. The foreign key is denoted also by *BId* in *Album* table and reference the primary key *BId* from *Band* table.

The code to create the *Band* table follows, and it will be the same for all the options given to create *Album* table, and more exactly the Foreign Key constraint.

CREATE TABLE Band(BId INT PRIMARY KEY, Name VARCHAR(50) NOT NULL, OriginCountry VARCHAR(50), FormedYear INT)	

Moreover, all the options presented will generate the same relationship 1:n between the tables Band and Album (through the primary key from Band table and foreign key from Album table).



To create a new table with a foreign key field / attribute / column that references another table, it will be used the keyword **FOREIGN KEY REFERENCES** after the name and type of the field / attribute / column, followed then, by the name of the referenced table and its primary key, in parentheses.

Option 1 to create the Foreign Key constraint (in CREATE TABLE statement – new table):

CREATE TABLE Album(Aid INT PRIMARY KEY IDENTITY, Title VARCHAR(50) NOT NULL, ReleasedYear INT, ProducerName VARCHAR(50), BId INT FOREIGN KEY REFERENCES Band(BId))	BId (primary key) from Band table has the same type as BId (foreign key) from Album table (INT type). The field *BId* from *Album* table is the foreign key that indicates the value stored in the field *BId* from *Band* table. After *FOREIGN KEY REFERENCES* is the primary key table and the primary key field: Band(BId).

Option 2 to create the Foreign Key constraint (in CREATE TABLE statement – new table):

CREATE TABLE Album(Aid INT PRIMARY KEY IDENTITY, Title VARCHAR(50) NOT NULL, ReleasedYear INT, ProducerName VARCHAR(50),	BId (primary key) from Band table has the same type as BId (foreign key) from Album table (INT type). The field *BId* from *Album* table is the foreign key that indicates the

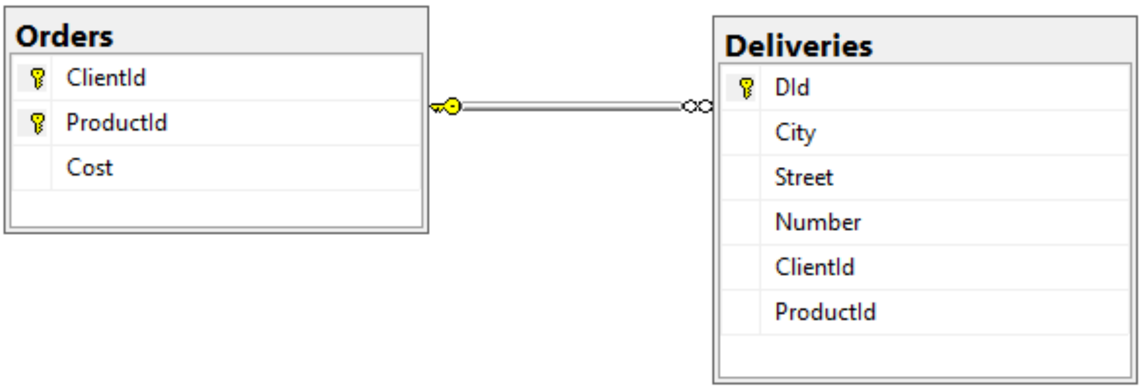
BId INT, FOREIGN KEY (BId) REFERENCES Band(BId))	value stored in the field *BId* from *Band* table. *BId* - the foreign key - is declared as an INT type field and then it is assign to it the Foreign Key constraint: *FOREIGN KEY (BId) REFERENCES* the primary key table and the primary key field: Band(BId).

Option 3 to create the Foreign Key constraint (in CREATE TABLE statement – new table):

CREATE TABLE Album(Aid INT PRIMARY KEY IDENTITY, Title VARCHAR(50) NOT NULL, ReleasedYear INT, ProducerName VARCHAR(50), BId INT, CONSTRAINT fk_Album_BId FOREIGN KEY (BId) REFERENCES Band(BId))	BId (primary key) from Band table has the same type as BId (foreign key) from Album table (INT type). The field *BId* from *Album* table is the foreign key that indicates the value stored in the field *BId* from *Band* table. *BId* - the foreign key - is declared as an INT type field and then it is assign to it the Foreign Key constraint: *FOREIGN KEY (BId) REFERENCES* the primary key table and the primary key field: Band(BId). The Foreign Key constraint also is denoted as a *CONSTRAINT* with a given name (fk_Album_BId), making possible the modification / removement of it from the table.

Option 4 to create the Foreign Key constraint (in CREATE TABLE statement – new table): Create a foreign key to a composed primary key (a primary key that is formed of multiple fields).

```
-- Primary Key table
CREATE TABLE Orders(
ClientId INT,
ProductId INT,
Cost FLOAT,
CONSTRAINT pk_Orders PRIMARY KEY(ClientId, ProductId))

-- Foreign Key table
CREATE TABLE Deliveries(
DId INT PRIMARY KEY IDENTITY,
City VARCHAR(50),
Street VARCHAR(50),
Number INT,
ClientId INT NOT NULL,
ProductId INT NOT NULL,
CONSTRAINT fk_Delivery
FOREIGN KEY (ClientId, ProductId) REFERENCES Orders (ClientId, ProductId))
```



Other possible option to add a Foreign Key constraint is in an already created / existing table, to add it with the help of the ALTER TABLE statement. It could be added simply with or without a name for the given constraint.

Option 5a to create the Foreign Key constraint (in ALTER TABLE statement – existing table – without a name for the constraint):

```
CREATE TABLE Album(
Aid INT PRIMARY KEY IDENTITY,
Title VARCHAR(50) NOT NULL,
ReleasedYear INT,
ProducerName VARCHAR(50),
BId INT NOT NULL)

ALTER TABLE Album
ADD FOREIGN KEY (BId) REFERENCES Band(BId)
```

Option 5b to create the Foreign Key constraint (in ALTER TABLE statement – existing table – without a name for the constraint):

```
CREATE TABLE Album(
Aid INT PRIMARY KEY IDENTITY,
Title VARCHAR(50) NOT NULL,
ReleasedYear INT,
ProducerName VARCHAR(50),
BId INT NOT NULL)

ALTER TABLE Album
ADD CONSTRAINT fk_Album_BId
ADD FOREIGN KEY (BId) REFERENCES Band(BId)
```

A full example to create a database

In this part, we will present an example in which is created a database, with its full source code and the generated diagram from SQL Server.

Database Description: Create a database to manage the car races. The database will store data about the races in which are participating the pilots with their cars.
a) The entities of interest to the problem domain are: *Races, Cars, Pilots* and *Locations*.
b) Each race has a title, a start and an end point and take place at a specified location. The location has just the name of the country and of the city.
c) Each car has a brand, a model and a color.
d) Each pilot has a name, a surname, a gender, a date of birthday, and a list of races with the win prize. The win prize is given by the value of the prize (e.g. 1000 $).
e) Corresponding to each car and each race are given also the arrival date and the fee paid.

Database Requirement: Write an SQL script that creates the corresponding relational data model and generate the database diagram.

Solution:
Due to the given Database Description, we will have the following relationships between the tables involved:

Race - Car – m:n (many-to-many)
Race - Pilot – m:n (many-to-many)
Location - Race – 1:n (one-to-many)

In the source code that follows will be also considered the following elements:

- Each table needs to have one primary key defined for it. No more, no less. The primary key could be a single field / attribute / column, or a pair of multiple fields.
- A primary key can be defined as a pair of multiple fields, fields that could be foreign keys in other tables – usually, this is the best practice to create the intermediate table from the relationship m:n, related to how the primary key is defined.
- A table can have one or multiple foreign key(s), that establish the relationships with one or multiple tables.
- The NOT NULL constraint will not allow the assignment of a NULL value in the corresponding field to which it was defined.
- The IDENTITY property set by default a numbering mechanism, that is autogenerated (1, 2, 3, ...) for the field to which it is assigned.
- The DEFAULT constraint set and give the specified value of the corresponding field to which it was assigned, if no other value was introduced there.
- The CHECK constraint restricts the set of values that could be introduced for the field involved, to only the ones mentioned by it.
- The UNIQUE constraint does not allow to introduce duplicate values for the field involved.

The order in which the tables are created is very important. Always, have to be created first the tables that have only primary key and only after that the tables with foreign key(s) (like, in the relationship parent table – child table (1:n)).

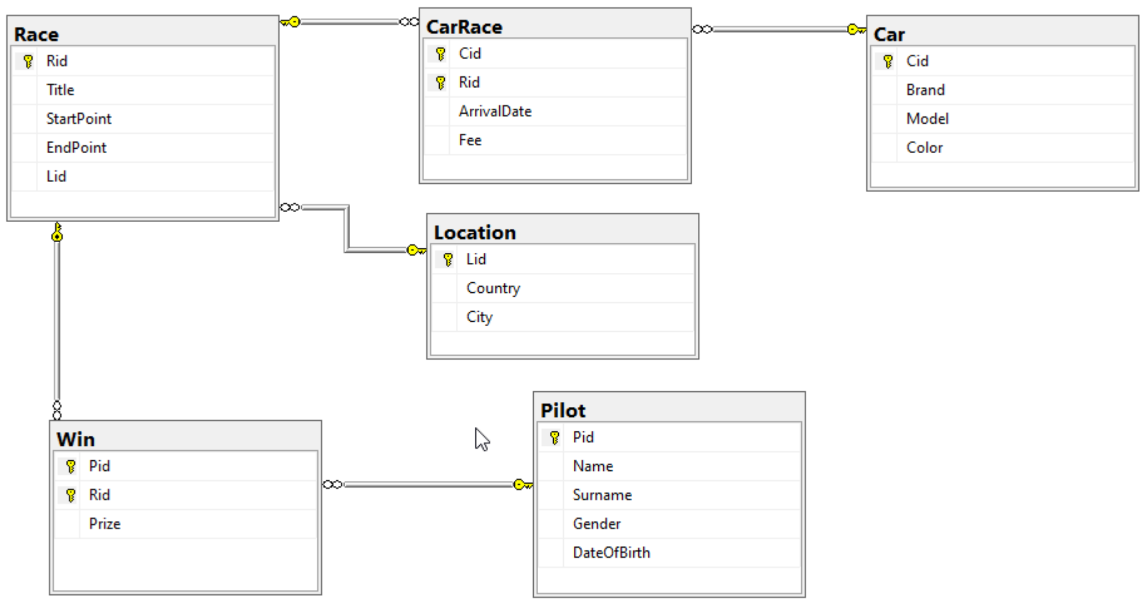
The relationship 1:n is the most important one and the basic one.

The relationship m:n (many-to-many) could be implemented with the help of an intermediate table, such that the relationship m:n is "split" it into two relationships 1:n.

Source code to generate the diagram follows:

```
create table Location(
Lid int primary key identity,
Country varchar(50) DEFAULT 'Romania',
City varchar(50))

create table Race(
Rid int primary key identity,
Title varchar(50) NOT NULL,
StartPoint varchar(50),
EndPoint varchar(50),
Lid int foreign key references Location(Lid))

create table Car(
Cid int primary key identity,
Brand varchar(50) NOT NULL,
Model varchar(50) UNIQUE,
Color varchar(50) CHECK (Color IN ('Blue', 'Red', 'Gray')))

create table Pilot(
Pid int primary key identity,
Name varchar(50) NOT NULL,
Surname varchar(50),
Gender varchar(30) CHECK(Gender IN ('Male', 'Female', 'Other')),
DateOfBirth date)

create table Win( -- intermediate table for relationship m:n Pilot-Race
Pid int foreign key references Pilot(Pid),
Rid int foreign key references Race(Rid),
Prize money,
CONSTRAINT pk_Win PRIMARY KEY(Pid, Rid))

create table CarRace( -- the intermediate table for relationship m:n Car-Race
Cid int foreign key references Car(Cid),
Rid int foreign key references Race(Rid),
ArrivalDate date,
Fee float,
CONSTRAINT pk_CarRace PRIMARY KEY(Cid, Rid))
```

Diagram:



References:
Elena-Maria Mărginean, Emilia-Loredana Pop, Create a table with a Foreign Key Constraint in SQL Server, Concurs Național de Referate și Comunicări Științifice "Ștefan Procopiu", Ediția XXX, CAEN 2022-2023.