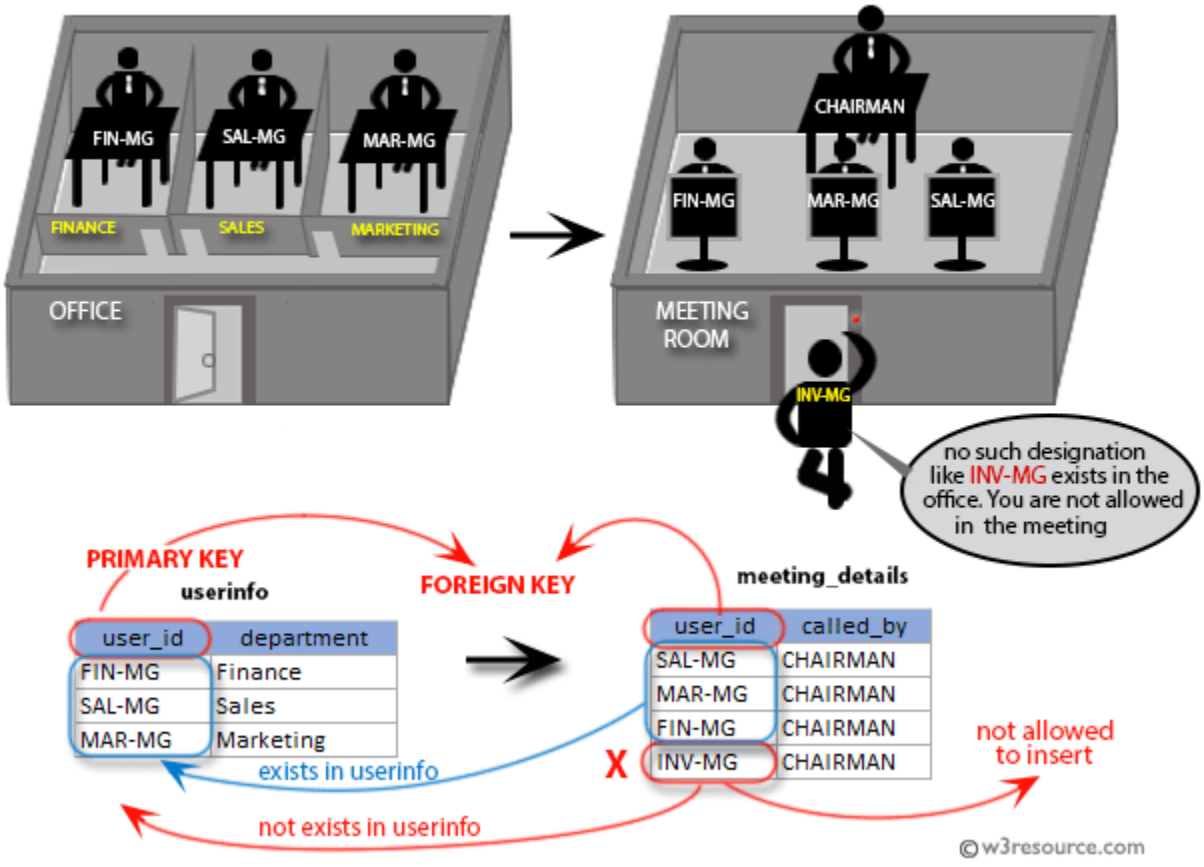
FOREIGN KEY

The SQL FOREIGN KEY CONSTRAINT is used to ensure the referential integrity of the data in one table to match values in another table.

The FOREIGN KEY CONSTRAINT is a column or list of columns which points to the PRIMARY KEY of another table.

The main purpose of FOREIGN KEY is, only those values will appear which are present in the primary key table.



For each row in the referencing table(the table contains the FOREIGN KEY), the foreign key must match an existing primary key in the referenced table(the table contains the PRIMARY KEY). This enforcement of FOREIGN KEY called the Referential Integrity.

The structure and data type of PRIMARY KEY and FOREIGN KEY must be same.

The values of the FOREIGN KEY columns in each row of the referencing table have to match with the values of the corresponding primary key columns of a row in the referenced table.

Syntax:

```
CREATE TABLE <table_name>(
column1    data_type[(size)] ,
column2    data_type[(size)] ,
constraint(constraint_name)
FOREIGN KEY  [column1,column2...]
REFERENCES [primary_key_table] (column_list_of_primary_key_table) ...);
```

Parameters:

Name	Description
table_name	The name of the table where data is stored.
column1,column2	Name of the columns of a table.
data_type	Is char, varchar, integer, decimal, date and more.
size	Maximum length of the column of a table.
constraint	Is a key word. This key word is optional.
constraint_name	Is a constraint name defined by user.
primary_key_table	Table where primary key resides.
column_list_of_primary_key_table	List of columns which makes primary key for a table.

Example:

Suppose, we have a table 'agents', that includes all agents data, and we are going to create another table named 'customer1', that includes all customers records. The columns and data types for both the tables have shown bellow.

The constraint here is that all customers must be associated with an agent that is already in the 'agents' table. In this case, an SQL FOREIGN KEY CONSTRAINT should be created with the 'customer1' table which is related to the SQL PRIMARY KEY CONSTRAINT of the 'agents' table.

Now, we can ensure that all customers in the 'customer1' table are related to an agent in the 'agents' table. In other words, the 'customer1' table can not contain information of any agent who is not on the 'agents' table.

agents

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
agent_code	char	6		No	PRIMARY KEY
agent_name	char	40		No	
working_area	char	35		Yes	
commission	decimal	10	2	Yes	

phone_no	char	17	Yes

customer1

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
cust_code	char	6		No	PRIMARY KEY
cust_name	char	25		Yes	
cust_city	char	25		Yes	
agent_code	char	6		Yes	FOREIGN KEY

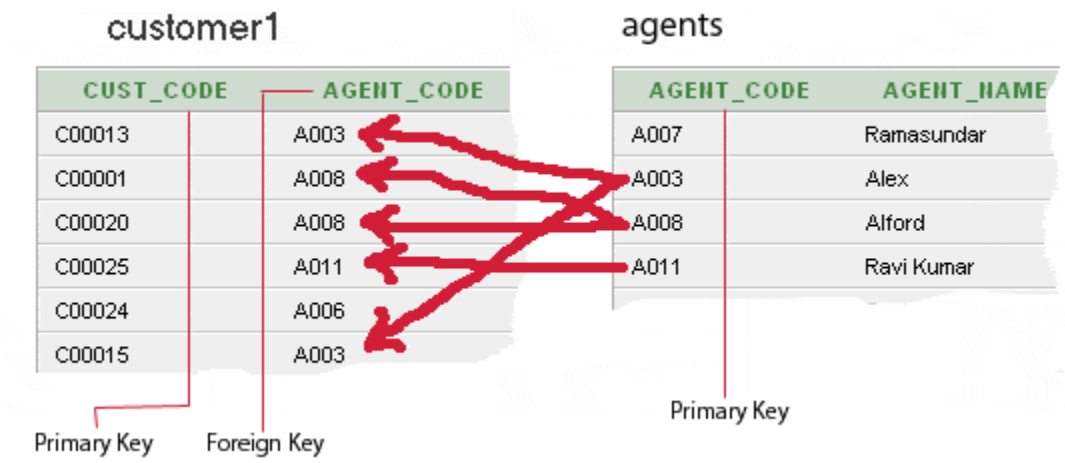
the following SQL statement can be used :

SQL Code:

```
CREATE TABLE customer1(
cust_code char(6) NOT NULL PRIMARY KEY,
cust_name char(25),
cust_city char(25),
agent_code char(6),
FOREIGN KEY(agent_code)
REFERENCES agents (agent_code)
) ;
```

Copy

Pictorial representation



SQL CREATE TABLE with FOREIGN KEY in column constraint

In the following topic, we are going to discuss the usage of FOREIGN KEY CONSTRAINT without using the 'foreign key' keyword.

Example:

To create a table which contains the following field names and data types.

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
cust_code	char	6		No	PRIMARY KEY
cust_name	char	25		Yes	
cust_city	char	25		Yes	
agent_code	decimal	6		Yes	

The table contains a PRIMARY KEY CONSTRAINT on 'cust_code' and a FOREIGN KEY on 'agent_code' without using the FOREIGN KEY key word.

The 'agent_code' in 'agents' table are unique.

Only those 'agent_code' which are present in 'agents' table will appear in 'mytest' table because reference column is 'agent_code' of 'agents' table.

the following SQL statement can be used :

SQL Code:

```
CREATE TABLE mytest(
cust_code char(6) NOT NULL PRIMARY KEY,
cust_name char(25),
cust_city char(25),
agent_code char(6)
REFERENCES agents(agent_code));
```

To see the structure of the created table:

SQL Code:

```
DESCRIBE mytest;
```

Output:

Object Type **TABLE** Object **MYTEST**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MYTEST	CUST_CODE	Char	6	-	-	1	-	-	-
	CUST_NAME	Char	25	-	-	-	✓	-	-
	CUST_CITY	Char	25	-	-	-	✓	-	-
	AGENT_CODE	Char	6	-	-	-	✓	-	-
									1 - 4

SQL CREATE TABLE using FOREIGN KEY CONSTRAINT without specifying PRIMARY KEY column(s)

In the following topic, we are going to discuss, how FOREIGN KEY CONSTRAINT can be used without specifying the primary key column(s).

Example:

To create a table containing the following field names and data types:

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
cust_code	char	6		No	PRIMARY KEY
cust_name	char	25		Yes	
cust_city	char	25		Yes	
agent_code	decimal	6		Yes	

The table contains a PRIMARY KEY CONSTRAINT on 'cust_code' and a FOREIGN KEY on 'agent_code' without specifying the PRIMARY KEY column -

The 'agent_code' in 'agents' table are unique.

Only those 'agent_code' which are present in 'agents' table will appear in 'mytest' table because reference column is 'agent_code' of 'agents' table.

the following SQL statement can be used:

SQL Code:

```
CREATE TABLE mytest(
cust_code char(6) NOT NULL PRIMARY KEY,
cust_name char(25),
cust_city char(25),
agent_code char(6)
REFERENCES agents);
```

To see the structure of the created table:

SQL Code:

```
DESCRIBE mytest;
```
Copy

Output:

Object Type **TABLE** Object **MYTEST**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MYTEST	CUST_CODE	Char	6	-	-	1	-	-	-
	CUST_NAME	Char	25	-	-	-	✓	-	-
	CUST_CITY	Char	25	-	-	-	✓	-	-
	AGENT_CODE	Char	6	-	-	-	✓	-	-

1 - 4

SQL CREATE TABLE using FOREIGN KEY CONSTRAINT with PRIMARY KEY column list

In the following topic, it is going to be discussed that, how SQL FOREIGN KEY CONSTRAINT is used with primary key column list in a CREATE TABLE statement.

Example:

To create a table which contains the following field names and data types -

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
cust_code	char	6		No	PRIMARY KEY
cust_name	char	40		No	
cust_city	char	35		Yes	
working_area	char	35		Yes	
cust_country	char	20		Yes	
grade	decimal	4	0	Yes	
opening_amt	decimal	12	2	Yes	
receive_amt	decimal	12	2	Yes	
payment_amt	decimal	12	2	Yes	
outstanding_amt	decimal	12	2	Yes	
phone_no	char	17		Yes	
agent_code	char	7		Yes	FOREIGN KEY

The table contains a PRIMARY KEY CONSTRAINT on 'cust_code' and a FOREIGN KEY on 'agent_code'.

The 'agent_code' in 'agent1' table are unique,

Only those 'agent_code' which are present in 'agnet1' table will appear in 'mytest' table because reference column is 'agent_code' of 'agnet1' table,

the following SQL statement can be used:

SQL Code:

```
CREATE TABLE mytest(
cust_code char(6) NOT NULL PRIMARY KEY,
cust_name char(40) NOT NULL,
cust_city char(35),
working_area char(35),
cust_country char(20),
grade decimal(4,0),
opening_amt decimal(12,2),
receive_amt decimal(12,2),
payment_amt decimal(12,2),
outstanding_amt decimal(12,2),
phone_no char(17),
agent_code char(6),
FOREIGN KEY(agent_code)
REFERENCES agent1(agent_code),
UNIQUE(cust_code,agent_code));
```

To see the structure of the created table:

SQL Code:

```
DESCRIBE mytest;
```

Copy

Output:

Object Type **TABLE** Object **MYTEST**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MYTEST	CUST_CODE	Char	6	-	-	1	-	-	-
	CUST_NAME	Char	40	-	-	-	-	-	-
	CUST_CITY	Char	35	-	-	-	✓	-	-
	WORKING_AREA	Char	35	-	-	-	✓	-	-
	CUST_COUNTRY	Char	20	-	-	-	✓	-	-
	GRADE	Number	-	4	0	-	✓	-	-
	OPENING_AMT	Number	-	12	2	-	✓	-	-
	RECEIVE_AMT	Number	-	12	2	-	✓	-	-
	PAYMENT_AMT	Number	-	12	2	-	✓	-	-
	OUTSTANDING_AMT	Number	-	12	2	-	✓	-	-
	PHONE_NO	Char	17	-	-	-	✓	-	-
	AGENT_CODE	Char	6	-	-	-	✓	-	-

1 - 12

SQL CREATE TABLE using FOREIGN KEY on more than one column with PRIMARY KEY column list

In the following topic, we are going to discuss, how SQL FOREIGN KEY CONSTRAINT can be used on more than one columns with primary key column list in a CREATE TABLE statement.

Example:

To create a table which contains the following field names and data types -

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
ord_num	decimal	6		No	PRIMARY KEY
ord_amount	decimal	12	2	Yes	
advance_amount	decimal	12	2	No	
ord_date	date			No	
cust_code	char	6		No	FOREIGN KEY
agent_code	char	6		No	FOREIGN KEY
ord_description	char	60		No	

The table contains a PRIMARY KEY CONSTRAINT on 'ord_num' and a FOREIGN KEY in a combination of 'cust_code' and 'agent_code' column.

The 'cust_code' and 'agent_code' combination in 'customer1' table are unique.

Only those 'cust_code' and 'agent_code' combination which are present in 'customer1' table will appear in 'mytest' table because reference columns are 'cust_code' and 'agent_code' combination of 'customer1' table.

the following SQL statement can be used:

SQL Code:

```
CREATE TABLE mytest(
ord_num decimal(6) NOT NULL PRIMARY KEY,
ord_amount decimal(12,2),
advance_amount decimal(12,2) NOT NULL,
ord_date date NOT NULL,
cust_code char(6) NOT NULL,
agent_code char(6) NOT NULL,
ord_description char(60) NOT NULL,
FOREIGN KEY(cust_code,agent_code)
REFERENCES customer1 (cust_code,agent_code));
```

To see the structure of the created table:

SQL Code:

```
DESCRIBE mytest;
```
Copy

Output:

Object Type **TABLE** Object **MYTEST**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MYTEST	ORD_NUM	Number	-	6	0	1	-	-	-
	ORD_AMOUNT	Number	-	12	2	-	✓	-	-
	ADVANCE_AMOUNT	Number	-	12	2	-	-	-	-
	ORD_DATE	Date	7	-	-	-	-	-	-
	CUST_CODE	Char	6	-	-	-	-	-	-
	AGENT_CODE	Char	6	-	-	-	-	-	-
	ORD_DESCRIPTION	Char	60	-	-	-	-	-	-
									1 - 7

SQL CREATE TABLE by referring FOREIGN KEY to own table

In the following topic, we are going to discuss, how SQL FOREIGN KEY CONSTRAINT can be used to refer its own table in a CREATE TABLE statement.

Example:

To create a table which contains the following fields and data types -

Field Name	Data Type	Size	Decimal Places	NULL	Constraint
cust_code	char	6		No	PRIMARY KEY
cliant_name	char	40		No	UNIQUE
cliant_city	char	35		No	
supp_code	date	6		No	

The table contains a PRIMARY KEY on 'cust_code' and a FOREIGN KEY on 'supp_code' where both 'cust_code' and 'supp_code' belong to the 'mytest' table. To achieve the above, the following SQL statement can be used:

SQL Code:

```
CREATE TABLE mytest(
cust_code char(6) NOT NULL PRIMARY KEY,
cliant_name char(40) NOT NULL UNIQUE,
cliant_city char(35),
supp_code char(6) REFERENCES mytest);
```

To see the structure of the created table:

SQL Code:

```
DESCRIBE mytest;
```

Output:

Object Type **TABLE** Object **MYTEST**

Table	Column	Data Type	Length	Precision	Scale	Primary Key	Nullable	Default	Comment
MYTEST	CUST_CODE	Char	6	-	-	1	-	-	-
	CLIANT_NAME	Char	40	-	-	-	-	-	-
	CLIANT_CITY	Char	35	-	-	-	✓	-	-
	SUPP_CODE	Char	6	-	-	-	✓	-	-
									1 - 4