one-to-many relationship

one-to-many relationship

A *one-to-many* relationship between two tables means that a row in one table can have zero or more row in the table on the other side of their relationship.

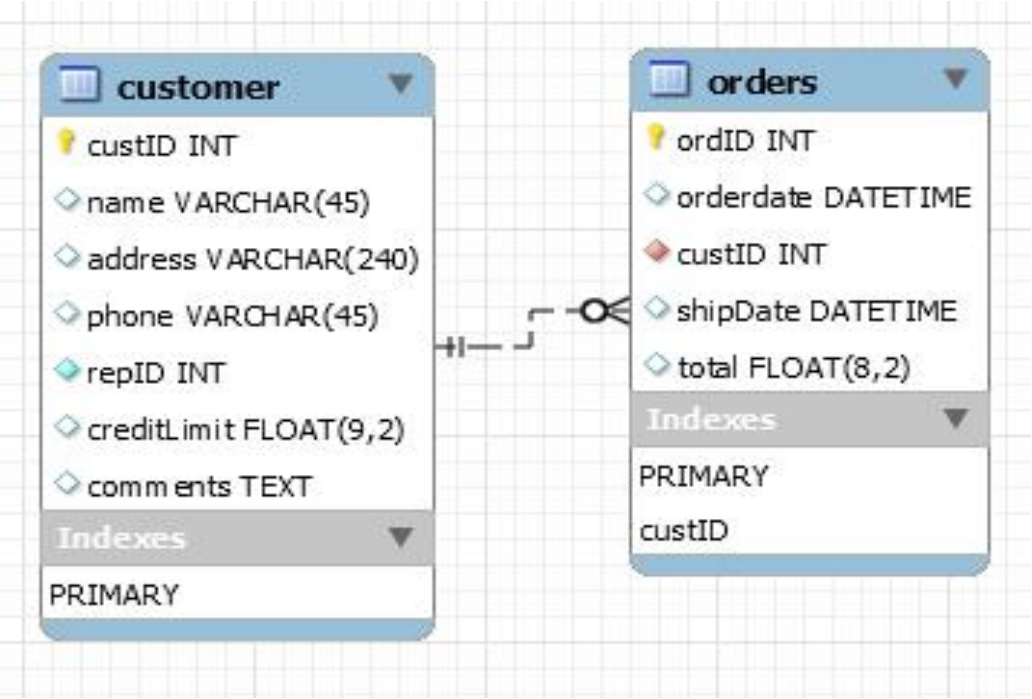
a *one-to-many* relationship is a type of cardinality that refers to the relationship between two entities ***R*** and ***S*** in which an element of ***R*** may be linked to many elements of ***S***, but a member of ***S*** is linked to only one element of ***R***.

Customer-1		Order-1	=	Customer-1	Order-1
Customer-2		Order-1		Customer-2	Order-1
Customer-3	*one-to-many relationship*	Order-2		Customer-2	Order-2
Customer-4		Order-1		Customer-3	Order-1
Customer-5		Order-2		Customer-3	Order-2
		Order-3		Customer-3	Order-3
		Order-1		Customer-4	Order-1

how to create one-to-many relationship

```
CREATE TABLE customer (
    custID INT PRIMARY KEY,
    name VARCHAR(45),
    address VARCHAR(240),
    phone VARCHAR(45),
    repID INT NOT NULL,
    creditLimit FLOAT(9,2),
    comments TEXT,
    constraint custid_zero CHECK(custID > 0)
);
```

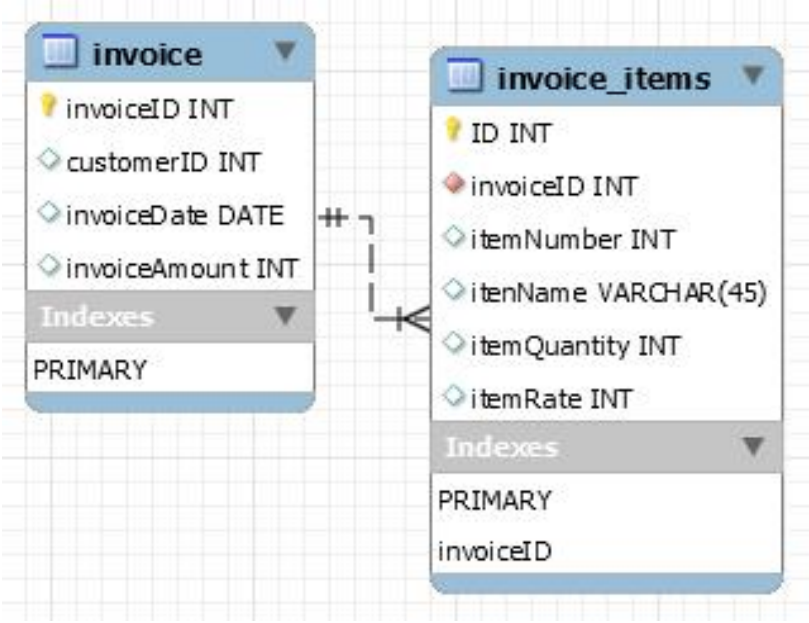
```
CREATE TABLE orders (
    ordID INT PRIMARY KEY,
    orderdate DATETIME,
    custID INT,
    shipDate DATETIME,
    total FLOAT(8,2),
    FOREIGN KEY(custID) REFERENCES customer(custID),
    constraint total_greater_zero CHECK(total >= 0)
);
```



how to create one-to-many relationship

```
CREATE TABLE invoice (
   invoiceID INT PRIMARY KEY,
   customerID INT,
   invoiceDate DATE,
   invoiceAmount INT
);
```

invoiceID	customerID	invoiceDate	invoiceAmount
1	235	2020-01-13	1750
2	235	2020-02-28	5000
3	778	2020-03-10	2000
4	778	2020-03-16	2300
NULL	NULL	NULL	NULL




```
CREATE TABLE invoice_items (
   invoiceID INT,
   itemID INT,
   itenName VARCHAR(45),
   itemQuantity INT,
   itemRate INT,
   PRIMARY KEY(invoiceID, itemID),
   FOREIGN KEY(invoiceID) REFERENCES
invoice(invoiceID)
);
```

```
CREATE TABLE invoice_items (
   invoiceID INT NOT NULL,
   itemID INT NOT NULL,
   itenName VARCHAR(45),
   itemQuantity INT,
   itemRate INT,
   UNIQUE(invoiceID, itemID),
   FOREIGN KEY(invoiceID) REFERENCES
invoice(invoiceID)
);
```

Composite key

Whenever a primary key consists of more than one attribute, it is known as a composite key. This key is also known as Concatenated Key.

For example,
in employee relations, we assume that an employee may be assigned multiple roles, and an employee may work on multiple projects simultaneously. So the primary key will be composed of all three attributes, namely Emp_ID, Emp_role, and Proj_ID in combination. So these attributes act as a composite key since the primary key comprises more than one attribute.