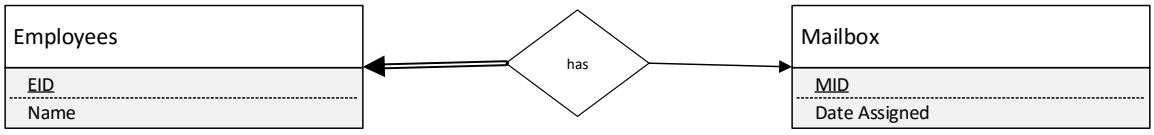
Converting Relationships to the Relational Model

Each of the key constraints is handled differently, as well as relationships with attributes as opposed to relationship without attributes.

Let's start with a 1:1 relationship. Using this example:



This defines the following:
Every employee must one and only one mailbox
Every mailbox may have one and only one employee. It may also be NULL meaning it is currently unassigned.

The SQL to support this looks like:

```
CREATE TABLE Employee (
        EID     CHAR(20),
        Name  CHAR(30),
        MID    Integer NOT NULL,
        PRIMARY KEY (EID),
        FOREIGN KEY (MID) REFERENCES Mailbox)

CREATE TABLE Mailbox (
        MID    Integer,
        DateAssigned  DATE,
        EID     CHAR(20),
        PRIMARY KEY (MID),
        FOREIGN KEY (EID) REFERENCES Employee)
```

Note that the total participation from Employee is represented by the NOT NULL phrase on the foreign key MID. Because the participation from Mailbox is partial, the phrase is not used.

Next is the 1:Many relationship shown in the following:

A customer may have 0 to many orders, an order has 1 and only one customer.

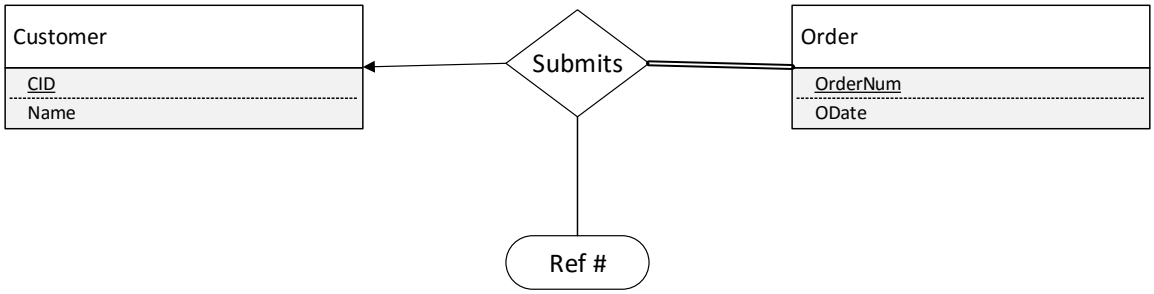
The SQL to represent this looks like:

```
CREATE TABLE Customer (
        CID     Integer,
        Name  CHAR(30),
        PRIMARY KEY (CID))

CREATE TABLE Order (
        OrderNum      Integer,
        ODate  DATE,
        CID     Integer NOT NULL,
        PRIMARY KEY(OrderNum),
        FOREIGN KEY (CID) REFERENCES Customer)
```

How would this change if there were attributes assigned to the relationship?

Let's look at the following:



Most of the time, it makes more sense to pull this into the many side of the 1:many. In the above situation you would have Ref number added to the Order. But let's say there is a structure there that you want to maintain for some reason. The SQL would look like the following:

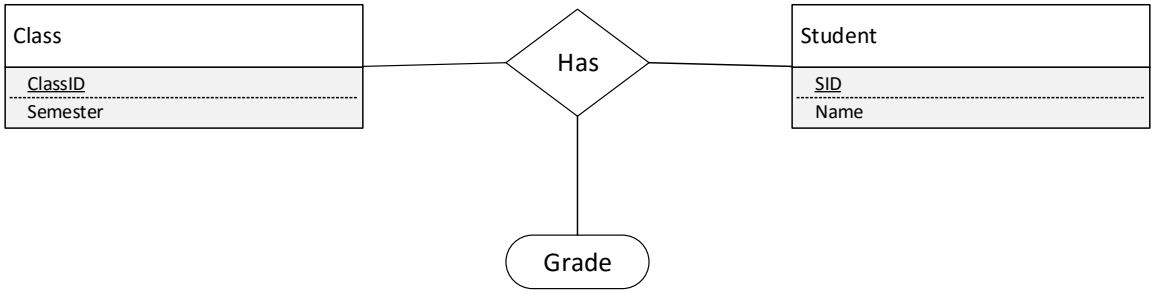
```
CREATE TABLE Customer (
        CID     Integer,
        Name  CHAR(30),
        PRIMARY KEY (CID))

CREATE TABLE Order (
        OrderNum      Integer,
        ODate  DATE,
        PRIMARY KEY(OrderNum))

CREATE TABLE Submits (
        CID             INTEGER NOT NULL,
        OrderNum      INTEGER,
        RefNum        INTEGER,
        PRIMARY KEY (OrderNum),
        FOREIGN KEY (OrderNum) REFERENCES Order,
        FOREIGN KEY CID REFENCES Customer)
```

Note that only OrderNum is required in the primary key due to the 1:many relationship.

Last we look at the many to many relationship.



The SQL to support this is:

```
CREATE TABLE Class (
        ClassID         INTEGER,
        Semester        CHAR (10),
        PRIMARY KEY (ClassID))

CREATE TABLE Students (
        SID             INTEGER,
        Name            CHAR(30),
        PRIMARY KEY (SID))

CREATE TABLE Enrolled (
        ClassID         INTEGER,
        SID             INTEGER,
        Grade           CHAR(2),
        PRIMARY KEY (ClassID, SID),
        FOREIGN KEY (ClassID ) REFERENCES Class,
        FOREIGN KEY (SID) REFERENCES Students)
```

Note that Enrolled requires both primary keys of the Entities associated with it in its primary key to support the many to many.

There are many issues associated with deleting records from Entities associated with a many to many relationship – these are outlined in the Referential Integrity example.