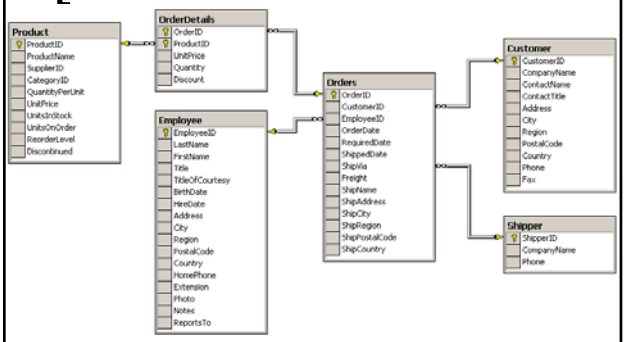
CGT 356

Lecture 17

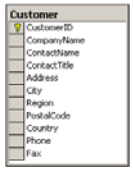
Lab 7

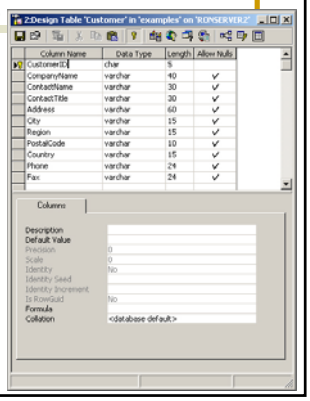
- Starting with the tables that do not have a foreign key…

Entity Relationship Diagram (ERD)



Customer Table

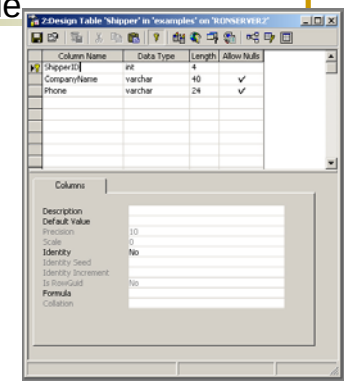




CREATE Customer

```
CREATE TABLE Customer(
CustomerID   char(5) CONSTRAINT CustPriKey PRIMARY KEY,
CompanyName  varchar(40),
ContactName  varchar(30),
ContactTitle varchar(30),
Address      varchar(60),
City         varchar(15),
Region       varchar(15),
PostalCode   varchar(10),
Country      varchar(15),
Phone        varchar(24),
Fax          varchar(24)
);
```

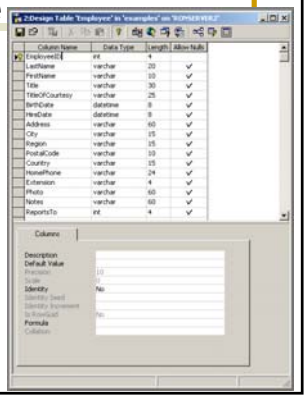
Shipper Table



CREATE Shipper

```
CREATE TABLE Shipper(
ShipperID   int CONSTRAINT ShipPriKey PRIMARY KEY,
CompanyName varchar(40),
Phone       varchar(24)
);
```

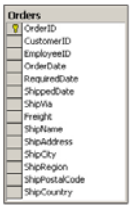
Employee Table

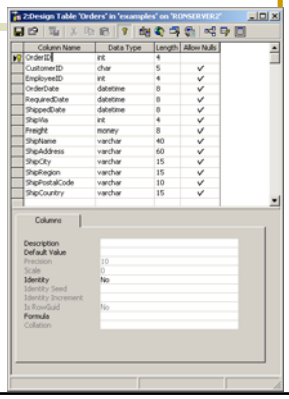


CREATE Employee

```
CREATE TABLE Employee(
EmployeeID      int CONSTRAINT EmpPriKey PRIMARY KEY,
LastName        varchar(20),
FirstName       varchar(10),
Title           varchar(30),
TitleOfCourtesy varchar(25),
BirthDate       datetime,
HireDate        datetime,
Address         varchar(60),
City            varchar(15),
Region          varchar(15),
PostalCode      varchar(10),
Country         varchar(15),
HomePhone       varchar(24),
Extension       varchar(4),
Photo           varchar(60),
Notes           varchar(60),
ReportsTo       int
);
```

Orders Table

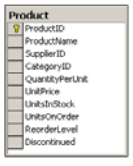


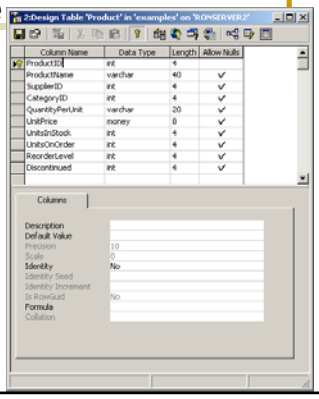


CREATE Orders

```
--Orders table must be plural because Order is a keyword in sql server
CREATE TABLE Orders(
OrderID         int      CONSTRAINT OrderPriKey    PRIMARY KEY,
CustomerID      char(5)  CONSTRAINT OrderFKCustID
                         REFERENCES Customer(CustomerID),
EmployeeID      int      CONSTRAINT OrderFKEmpID
                         REFERENCES Employee(EmployeeID),
OrderDate       datetime,
RequiredDate    datetime,
ShippedDate     datetime,
ShipVia         int      CONSTRAINT OrderFKShipVia
                         REFERENCES Shipper(ShipperID),
Freight         money,
ShipName        varchar(40),
ShipAddress     varchar(60),
ShipCity        varchar(15),
ShipRegion      varchar(15),
ShipPostalCode  varchar(10),
ShipCountry     varchar(15)
);
```

Product Table

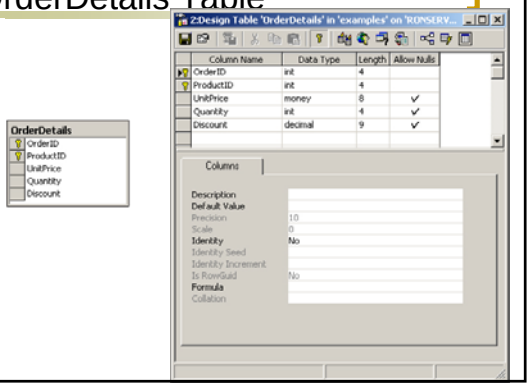




CREATE Product

```
CREATE TABLE Product(
ProductID       int CONSTRAINT ProductPriKey PRIMARY KEY,
ProductName     varchar(40),
SupplierID      int,
CategoryID      int,
QuantityPerUnit varchar(20),
UnitPrice       money,
UnitsInStock    int,
UnitsOnOrder    int,
ReorderLevel    int,
Discontinued    int
);
```

OrderDetails Table



CREATE OrderDetails

```
CREATE TABLE OrderDetails(
OrderID   int     CONSTRAINT OD_FK_OrderID
                  REFERENCES Orders(OrderID),
ProductID int     CONSTRAINT OD_FK_ProdID
                  REFERENCES Product(ProductID),
UnitPrice money,
Quantity  int,
Discount  decimal,
CONSTRAINT ODPriKey PRIMARY KEY(OrderID,
   ProductID)
);
```

Database Concepts

Primary Key – a designated field that cannot be null, which uniquely identifies a single record

Composite Primary Key – multiple selected fields that, combined, uniquely identify a single record, forming a Primary Key

References – a SQL keyword that creates a ***Relationship*** between two tables

Connectivity & Cardinality

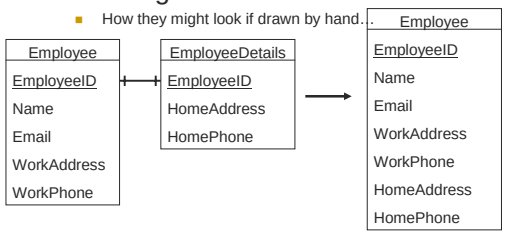
- Connectivity
  - The **connectivity** of a relationship describes the mapping of associated entity instances in the relationship.
  - The values of **connectivity** are "one" or "many".
- Cardinality
  - The **cardinality** of a relationship is the actual number of related occurences for each of the two entities.

Types of Connectivity

- Three types of connectivity
  (you could say, three types of relationships)
  - One to One
    - Not very useful
  - One to Many
    - Most common
  - Many to Many
    - Want to avoid this one at all costs

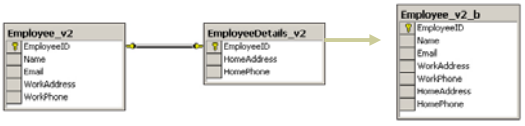
One to One

- These two tables can be combined into a single table.
  - How they might look if drawn by hand…



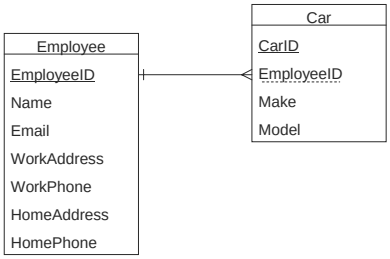
One to One

- The same tables from a SQL Server diagram view:



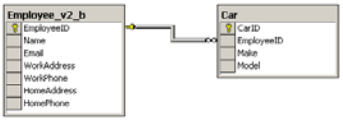
One to Many

- An employee can own many cars.
- A car can only have one owner.



One to Many

- The same tables from a SQL Server diagram view:



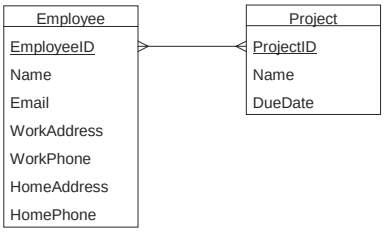
Many to Many

- For every record in Table1, there are multiple records in Table2.
- For every record in Table2, there are multiple records in Table1.

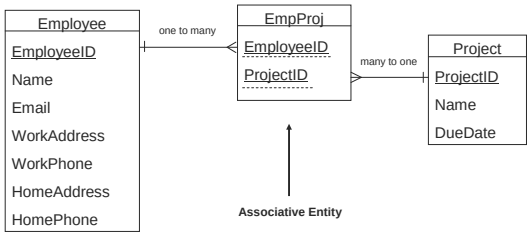
Associative Entity

- Most many to many relationships are broken up into one to many relationships.
- This causes the creation of an Associative Entity.
- An Associative Entity is an intermediate table created in between two tables that have a many-to-many relationship in order to create 2 one-to-many relationships

Many to Many



Associative Entity



Database Example

- Use this course as an example
- What are the main objects?
- How would we draw these out?

Database Example (cont.)

Further Thought

- What about accounting for the year they took the course?
- What about the semester?
- What if they retake the course?
- What about grades and attendance?

Database Example 2

- Image Management System
- What are the main objects?
- How would we draw these out?

Database Example 2 (cont.)