Relational DB Design

cs380
Oct 29

Revisit the Tower of Hanoi App

- Code for receiving/storing data was hideous
  - Receive data
  - Create a graph on HTML canvas using Javascript that is generated by PHP
- Too ugly to be allowed!!
- Broke it up into:
  - PHP: save data
  - PHP: retrieve data
  - HTML/JS: AJAX and make graph

file: aaasave.php

```php
<?php
$servername = "localhost";
$username = "gtstudent";
$password = "";
$dbname = "hanoi";
$conn = new mysqli($servername, $username, $password,
$dbname);
if ($conn->connect_error) {
    die("Connection failed: " . $conn-
>connect_error);
}

// Only accept POSTed data
$data1=$_POST['textdata'];
$data2=$_POST['ttime'];
$data3 = $_POST['observer'];
if (strcmp($data2, "3487")!=0) {
    $sql = "INSERT INTO timedata (actor, witness,
time) VALUES ('$data1', '$data3', '$data2');";
    //echo "$sql<br>";
    $conn->query($sql);
    }
// Tell the browser to load this page instead
header("Location: aaa.html", true, 301);
?>
```

Towers Graphing

file: aaa.html

```
<html>
    <head>
        <script src="../JQ/jquery-1.9.1.min.js"></script>
        <style>
            .cnvvas {
                width:90%;
                height:80%;
                border-top:5%;
                border-right:5%;
            }
        </style>
    </head>
    <body>
        <h1 style="height:8%;">Hanoi Speed Results</h1>
        <canvas id="eTTUV" class="cnvvas"></canvas>
        <script>
            // Setup NOT SHOWN
            /**
            * Use JQUery style of fetch to get data from server
            **/
            function getData() {
                let posted = $.post(baseURL+"aaadata.php");
                posted.done( function(data) {
                console.log(data);
                json = JSON.parse(data);
                makeGraph(json);
                });
            }
            // Graph Generation – not shown
```

file: aaadatag.php

```
<?php
function toSecs($v)
{
    $sss = 0;
    $xxx = explode(":", $v);
    if (count($xxx) > 1)
        return (double)$xxx[1];
    else
        return -1;
}
// DB connect and query not shown
$aaa = Array();
while (null != ($row = $result->fetch_assoc())) {
    $secs = toSecs($row["time"]);
    if ($secs > 0) {
        array_push($aaa, [substr($row['actor'],0,3), $secs] );
    }
}
echo json_encode($aaa);
?>
```

Graphing — Again and Again

file:aaa.html aaabar.html

```
    for(i=0; i<dta.length-1; i++) {
        strtx = inn+((i*xspc)/countt);
        strty = ysiz*0.1 + yspc-(((dta[i]-mnn)/range)*yspc);
        endx = inn+((i+1)*xspc/countt);
        endy = ysiz*0.1 + yspc-((dta[i+1]-mnn)/range)*yspc;
        writeLine(strtx, strty, endx, endy, "#FF00FF", 5);
    }

    // SWITCH TO A BAR GRAPH

   for(i=0; i<dta.length; i++) {
        strtx = inn+((i*xspc)/countt);
        endx = inn+((i+1)*xspc/countt);
        endy = ysiz*0.05 + yspc-((dta[i]-mnn)/range)*yspc;
        fillSpace(strtx, ysiz*0.95, endx, endy, "#FF00FF");
    }
```

Better yet ... use Google's graphing javascript
file: aaagoo.html

```
<html>
  <head>
   <script type="text/javascript" src="https://www.gstatic.com/charts/loader.js"
    <script src="../JQ/jquery-1.9.1.min.js"></script>
    <script type="text/javascript">
       google.charts.load('current', {'packages':['corechart', 'bar']});
       google.charts.setOnLoadCallback(getData);
        let chartReady=false;
        let baseURL = "http://comet.cs.brynmawr.edu/~gtowell/UC380/Lec12/";

       function makeGraph(json) {
        // Create the data table.
        var data = new google.visualization.DataTable();
        data.addColumn('string', 'User');
        data.addColumn('number', 'Time');
        data.addRows(json);
        // Set chart options
        var options = {'title':'Tower Times',
                       'legend':'none',
                       'backgroundColor':'antiquewhite'
                       };
       var chart = new google.visualization.BarChart(document.getElementById('ch
        chart.draw(data, options);
      }

  <body>
    <!--Div that will hold the pie chart-->
    <div id="chart_div" class="chartstyle"></div>
  </body>
</html>
```

Suppose a Database

users				
name	company	company_address	url1	url2
Joe	ABC	1 Work Lane	abc.com	xyz.com
Jill	XYZ	1 Job Street	abc.com	xyz.com

```
create table users1 (
        name varchar(20) NOT NULL,
        company varchar(20) not null,
        company_address varchar(20) not null,
        url1 varchar(20) not null,
        url2 varchar(20) not null
);
insert into users1 (name, company,
company_address, url1, url2) values ('Joe',
'ABC', '1 work Lane', 'abc.com', 'xyz.com');
insert into users1 (name, company,
company_address, url1, url2) values ('Jill',
'XYZ', 'i Job Street', 'abc.com', 'xyz.com');
```

https://phpbuilder.com/database-normalization-and-design-techniques/
https://phpbuilder.com/database-normalization-and-design-techniques-page-2/
https://phpbuilder.com/database-normalization-and-design-techniques-page-3/
https://phpbuilder.com/database-normalization-and-design-techniques-page-4/
https://phpbuilder.com/database-normalization-and-design-techniques-page-5/

DataBase Normalization

- First Normal Form — There are no multi-valued attributes
  - Eliminate repeating groups in individual tables.
    - e.g. the url1 & url2 fields
  - Create a separate table for each set of related data.
  - Identify each set of related data with a primary key.
    - So make a “userID” field to indicate that Joe in rows 1 and 2 is the same Joe

```
create table users1 (
       name varchar(20) NOT NULL,
       company varchar(20) not null,
       company_address varchar(20) not null,
       url varchar(20) not null,
);
insert into users1 (name, company, company_address,
url) values ('Joe', 'ABC', '1 work Lane',
'abc.com');
insert into users1 (name, company, company_address,
url) values ('Jill', 'XYZ', 'i Job Street',
'xyz.com');
insert into users1 (name, company, company_address,
url) values ('Joe', 'ABC', '1 work Lane',
'abc.com');
insert into users1 (name, company, company_address,
url) values ('Jill', 'XYZ', 'i Job Street',
'xyz.com');
```

users				
userId	name	company	company_address	url
1	Joe	ABC	1 Work Lane	abc.com
1	Joe	ABC	1 Work Lane	xyz.com
2	Jill	XYZ	1 Job Street	abc.com
2	Jill	XYZ	1 Job Street	xyz.com

DataBase Normalization

- Second Normal Form — Non-key fields must be dependent upon the entire key
  - Create separate tables for sets of values that apply to multiple records.
    - e.g. the URL field for first normal form
  - Relate these tables with a foreign key.
    - reluserid in URLs is a foreign key to userid in users

users			
userId	name	company	company_address
1	Joe	ABC	1 Work Lane
2	Jill	XYZ	1 Job Street

urls		
urlId	relUserId	url
1	1	abc.com
2	1	xyz.com
3	2	abc.com
4	2	xyz.com

DataBase Normalization

- Third Normal Form
  - Eliminate fields that do not depend on the key.
    - The company affiliation of Joe and Jill is NOT an attribute of those people and may be shared

users		
userId	name	relCompId
1	Joe	1
2	Jill	2

companies		
compId	company	company_address
1	ABC	1 Work Lane
2	XYZ	1 Job Street

urls		
urlId	relUserId	url
1	1	abc.com
2	1	xyz.com
3	2	abc.com
4	2	xyz.com

```
create table company3 (
    companyID int AUTO_INCREMENT PRIMARY KEY,
    company varchar(20),
    company_address varchar(20)
);
insert into company3 (company, company_address) values ('ABC', '1 Work Lane');
insert into company3 (company, company_address) values ('XYZ', '1 Job Street');
create table user3 (
    userid int AUTO_INCREMENT PRIMARY KEY,
    name varchar(20) NOT NULL,
    RelCompanyId int,
    FOREIGN KEY(RelCompanyId) REFERENCES company3(companyID)
);
insert into user3 (name, RelCompanyId) values ('Joe', 1);
insert into user3 (name, RelCompanyId) values ('Jill', 2);
create table urls3 (
    urlid int AUTO_INCREMENT PRIMARY KEY,
    relUserId int,
    url varchar(20),
    FOREIGN KEY(RelUserId) REFERENCES users2(userid)
);
```

DataBase Normalization

- The URLs table has info in it that is not an attribute of the URL. Namely the “reluser” column.
- This feels wrong.
  - Clean up by replacing the URL table with two:
    - URLs and URL_relations
- Fourth Normal Form
  - In a many-to-many relationship, independent entities can not be stored in the same table.

```
create table urls4 (
    urlid int AUTO_INCREMENT PRIMARY KEY,
    url varchar(20)
);
insert into urls4 (url) values ('abc.com');
insert into urls4 (url) values ('xyz.com');
create table url_rel4 (
    relationID int AUTO_INCREMENT PRIMARY KEY,
    RelURLid int,
    RelCompanyID int,
    FOREIGN KEY(RelURLid) REFERENCES urls4(urlid),
    FOREIGN KEY(RelCompanyId) REFERENCES company4(companyID)
);
insert into url_rel4 (RelURLid, relcompanyid) values (1, 1);
insert into url_rel4 (RelURLid, relcompanyid) values (1, 2);
insert into url_rel4 (RelURLid, relcompanyid) values (2, 1);
insert into url_rel4 (RelURLid, relcompanyid) values (2, 2);
```

users		
userId	name	relCompId
1	Joe	1
2	Jill	2

companies		
compId	company	company_address
1	ABC	1 Work Lane
2	XYZ	1 Job Street

urls	
urlId	url
1	abc.com
2	xyz.com

url_relations		
relationId	relatedUrlId	relatedUserId
1	1	1
2	1	2
3	2	1
4	2	2

Querying this database

users		
userId	name	relCompId
1	Joe	1
2	Jill	2

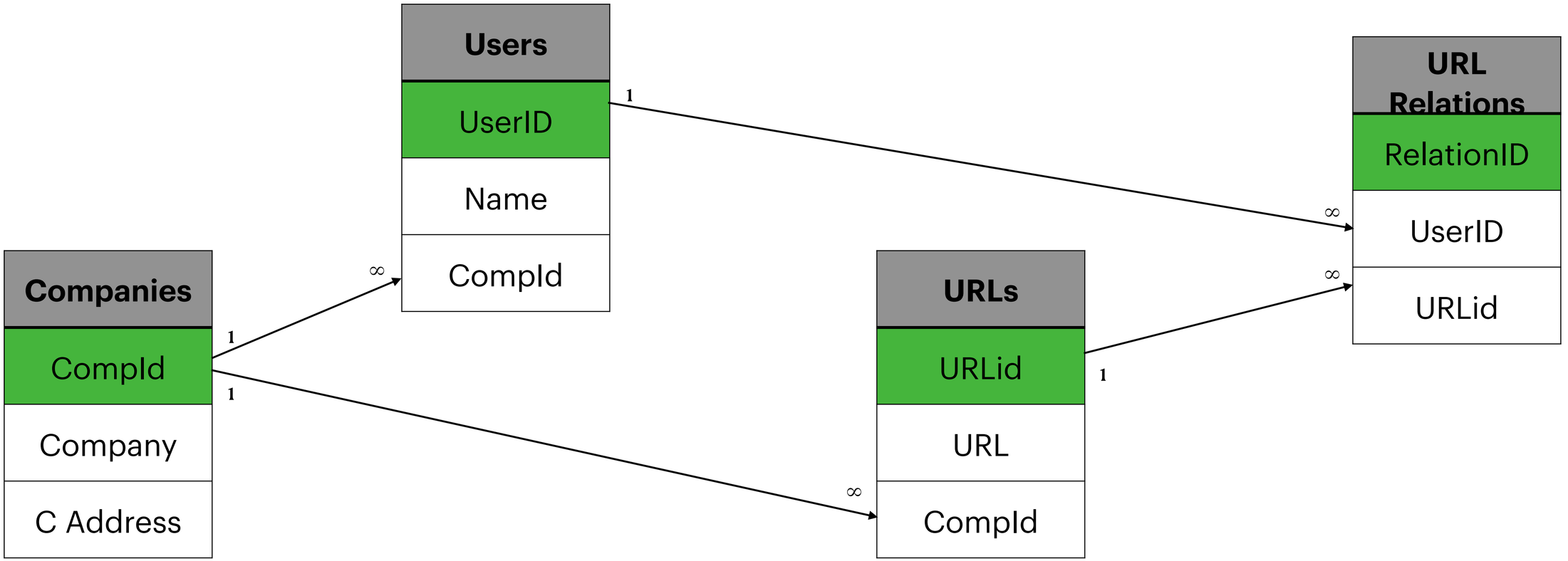
companies		
compId	company	company_address
1	ABC	1 Work Lane
2	XYZ	1 Job Street

urls	
urlId	url
1	abc.com
2	xyz.com

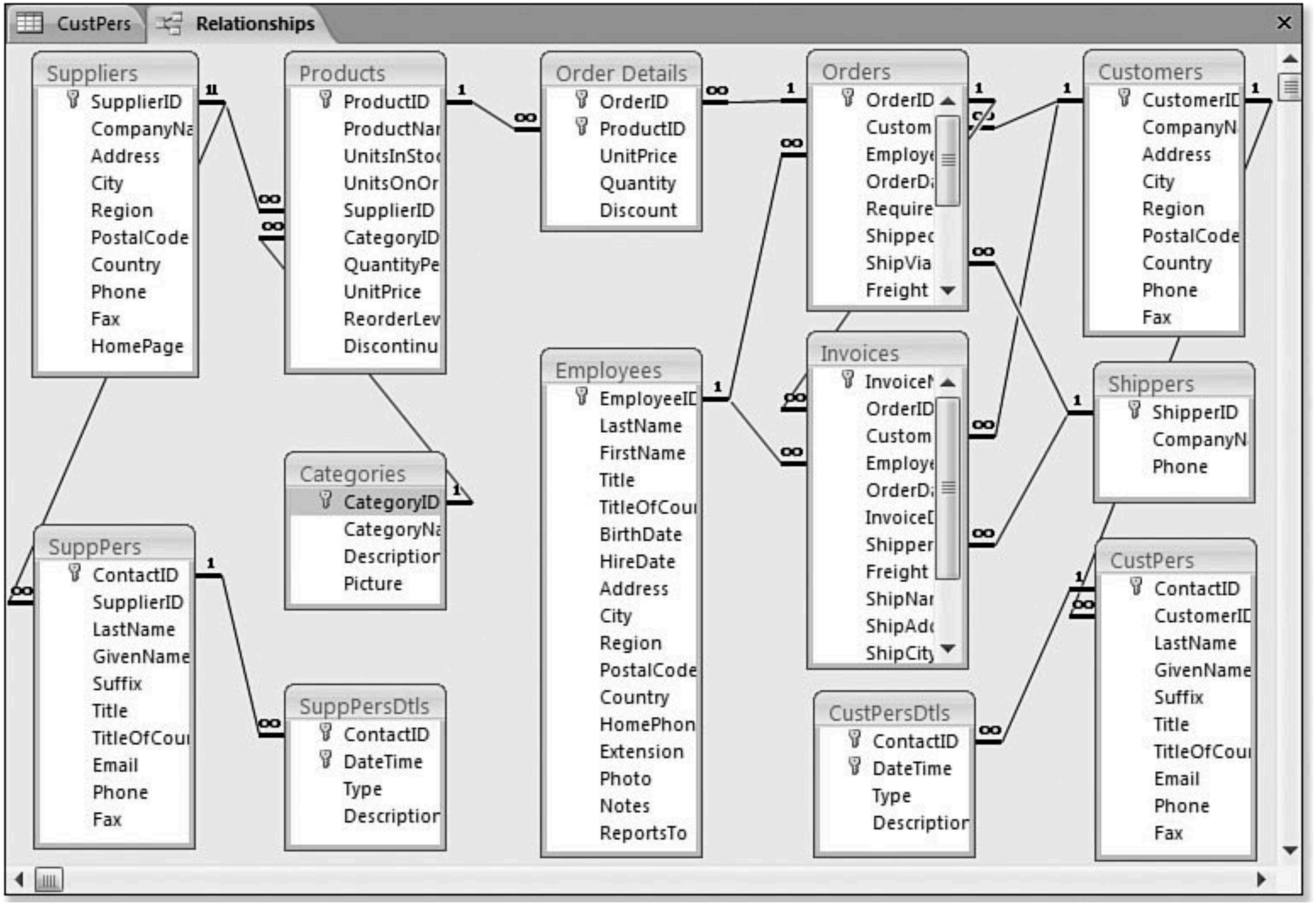
url_relations		
relationId	relatedUrlId	relatedUserId
1	1	1
2	1	2
3	2	1
4	2	2

- List all of Joe's urls
- List all employees of company 1
- List all employees of the company that Joe works for
- How many companies are in the database?

Representing a database



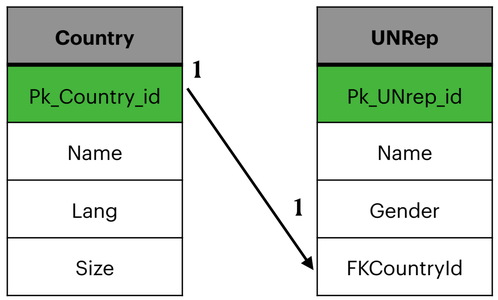
- Table Name at the top (grey background)
- Primary key next (green background)
- Arrows indicate foreign keys
- Numbers by arrows indicate type of relationship


CustPers
Relationships
Suppliers
SupplierID
CompanyNa
Address
City
Region
PostalCode
Country
Phone
Fax
HomePage
Products
ProductID
ProductNar
UnitsInStoc
UnitsOnOr
SupplierID
CategoryID
QuantityPe
UnitPrice
ReorderLev
Discontinu
Order Details
OrderID
ProductID
UnitPrice
Quantity
Discount
Orders
OrderID
Custom
Employe
OrderDa
Require
Shippec
ShipVia
Freight
Customers
CustomerID
CompanyN
Address
City
Region
PostalCode
Country
Phone
Fax
Employees
EmployeeID
LastName
FirstName
Title
TitleOfCour
BirthDate
HireDate
Address
City
Region
PostalCode
Country
HomePhon
Extension
Photo
Notes
ReportsTo
Invoices
InvoiceN
OrderID
Custom
Employe
OrderDa
InvoiceD
Shipper
Freight
ShipNar
ShipAdc
ShipCity
Shippers
ShipperID
CompanyN
Phone
Categories
CategoryID
CategoryNa
Description
Picture
SuppPers
ContactID
SupplierID
LastName
GivenName
Suffix
Title
TitleOfCour
Email
Phone
Fax
SuppPersDtls
ContactID
DateTime
Type
Description
CustPersDtls
ContactID
DateTime
Type
Description
CustPers
ContactID
CustomerID
LastName
GivenName
Suffix
Title
TitleOfCour
Email
Phone
Fax

Keys relation in enforced by DB

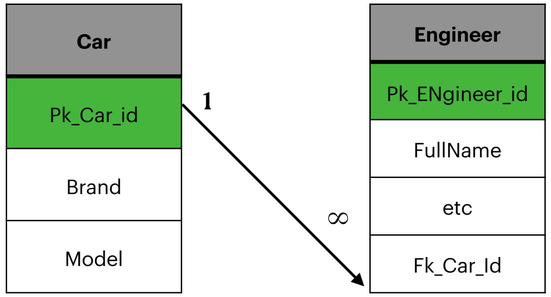
- One-to-One

```
CREATE TABLE Country
(
Pk_Country_Id INT PRIMARY
KEY,
Name VARCHAR(100),
Officiallang VARCHAR(100),
Size INT(11)
);

CREATE TABLE UNrep
(
Pk_UNrep_Id INT PRIMARY KEY,
Name VARCHAR(100),
Gender VARCHAR(100),
Fk_Country_Id INT UNIQUE,
FOREIGN KEY(Fk_Country_Id)
REFERENCES
Country(Pk_Country_Id)
);
```



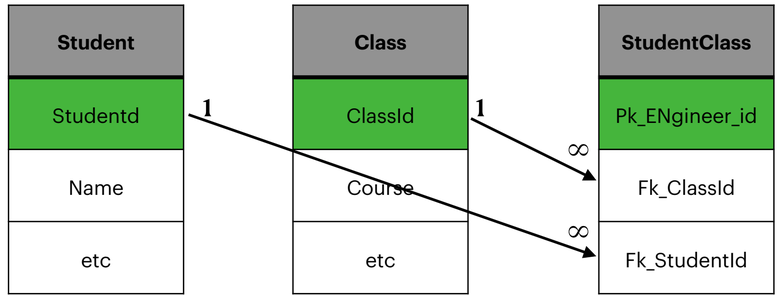
- One-to-Many

```
CREATE TABLE Car
(
Pk_Car_Id INT PRIMARY KEY,
Brand VARCHAR(100),
Model VARCHAR(100)
);

CREATE TABLE Engineer
(
Pk_Engineer_Id INT PRIMARY
KEY,
FullName VARCHAR(100),
MobileNo CHAR(11),
Fk_Car_Id INT,
FOREIGN KEY(Fk_Car_Id)
REFERENCES Car(Pk_Car_Id)
);
```



- Many-to-Many

```
CREATE TABLE Student(
StudentID INT(10),
Name VARCHAR(100),
PRIMARY KEY(studentID)
);

CREATE TABLE Class(
ClassID INT(10) PRIMARY KEY,
Course VARCHAR(100)
);

CREATE TABLE StudentClassRelation(
Fk_StudentID INT(15) NOT NULL,
Fk_ClassID INT(14) NOT NULL,
FOREIGN KEY (Fk_StudentID) REFERENCES
Student(StudentID),
FOREIGN KEY (Fk_ClassID) REFERENCES
Class(ClassID),
UNIQUE (StudentID, ClassID)
);
```



Database creation tips

- Keep data items atomic (e.g., first and last names are separate). Concatenating columns together later on-the-fly is generally easy, but separating them is not. (First Normal Form)
- Define the primary key first. Use a descriptive name (PARCELID, not ID)
- Use a single column for the primary key whenever possible; multi-column primary keys are appropriate for many-to-many relationships
- Use lookup tables rather than storing long values
  - Lookup tables are tables that have a key and a name ... for instance: car manufacturers, car models, ... They tend to be static
- Use numeric keys whenever possible (What about ZIP codes?)
- Avoid using multiple columns to represent a one-to-many relationship (e.g., columns such as CHILD1, CHILD2 in a table called PARENT rather than putting the children in a separate table. (First Normal Form)
- For readability, use the primary key name for foreign keys unless the same foreign key is used multiple times in the same table (e.g., state of work and state of residence for a person might both be foreign keys that reference a table of states)
- Do not include two columns whose values are linked together (e.g., county name and county ID) unless one of the columns is the primary key of the table (Third Normal Form)
- Avoid allowing NULL values in columns that have a discrete range of possible values (e.g., integers between 1 and 10, inclusive)
- Avoid using multiple tables with similar structures that represent minor variants on the same entity (e.g., putting Boston parcels and Cambridge parcels in separate tables).

Music Radio Station

- To decide what music to play next need to know
  - Music in library
  - Music played recently
  - Customer requests
  - Ratings