Relational DB Design
The logic of file layout in Android

cs380
Nov 5

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

Query to get all info from this database (i.e., userso table)?

```
create table users0 (
        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 users0 (name, company,
company_address, url1, url2) values ('Joe',
'ABC', '1 work Lane', 'abc.com', 'xyz.com');
insert into users0 (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 — 1NF

- 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

Query to get all info from this database (i.e., users1 table)?

DataBase Normalization — 2NF

- 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

```
create table users2 (
    userid int AUTO_INCREMENT PRIMARY KEY,
    name varchar(20) NOT NULL,
    company varchar(20) not null,
    company_address varchar(20) not null
);
insert into users2 (name, company, company_address)
values ('Joe', 'ABC', '1 work Lane');
insert into users2 (name, company, company_address)
values ('Jill', 'XYZ', 'i Job Street');

create table urls2 (
    urlid int AUTO_INCREMENT PRIMARY KEY,
    relUserId int,
    url varchar(20),
    FOREIGN KEY(RelUserId) REFERENCES users2(userid)
);
insert into urls2 (reluserid, url)
values (1, 'abc.com');
insert into urls2 (reluserid, url)
values (1, 'xyz.com');
insert into urls2 (reluserid, url)
values (2, 'abc.com');
insert into urls2 (reluserid, url)
values (2, 'xyz.com');
```

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

Query to get all info from this database (i.e., users2 and urls2 tables)?

DataBase Normalization — 3NF

- 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);
/** table urls3 is identical to tables urls2 **/
```

Query to get all info from this database (i.e., company3, users3, and urls3 tables)?

DataBase Normalization — 4NF

- 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.

```
/** company4 and user4 exactly follow company3 and user3 **/
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

Inserts into normalized DBs

- Auto increments id fields are really handy BUT
  - how do you know what they are when you insert into another table?
    - E.g.,

```
insert into user3 (name, RelCompanyId) values ('Joe', 1);
insert into user3 (name, RelCompanyId) values ('Jill', 2);
```

  - Need to do insert that uses a query!
    - Simplest version of insert using a query

```
INSERT INTO tableC (c1, c2, c3) SELECT dd1, dd2,dd3 FROM tableDD;
```

      - This violates RDB principle of only storing data once!
    - More useful
      -

```
insert into user4 (name, RelCompanyId) select 'Joe',company4.companyid
from company4 where company4.company='ABC';
```

Querying this database

users		
userId	name	relCompId
1	Joe	1
2	Jill	2

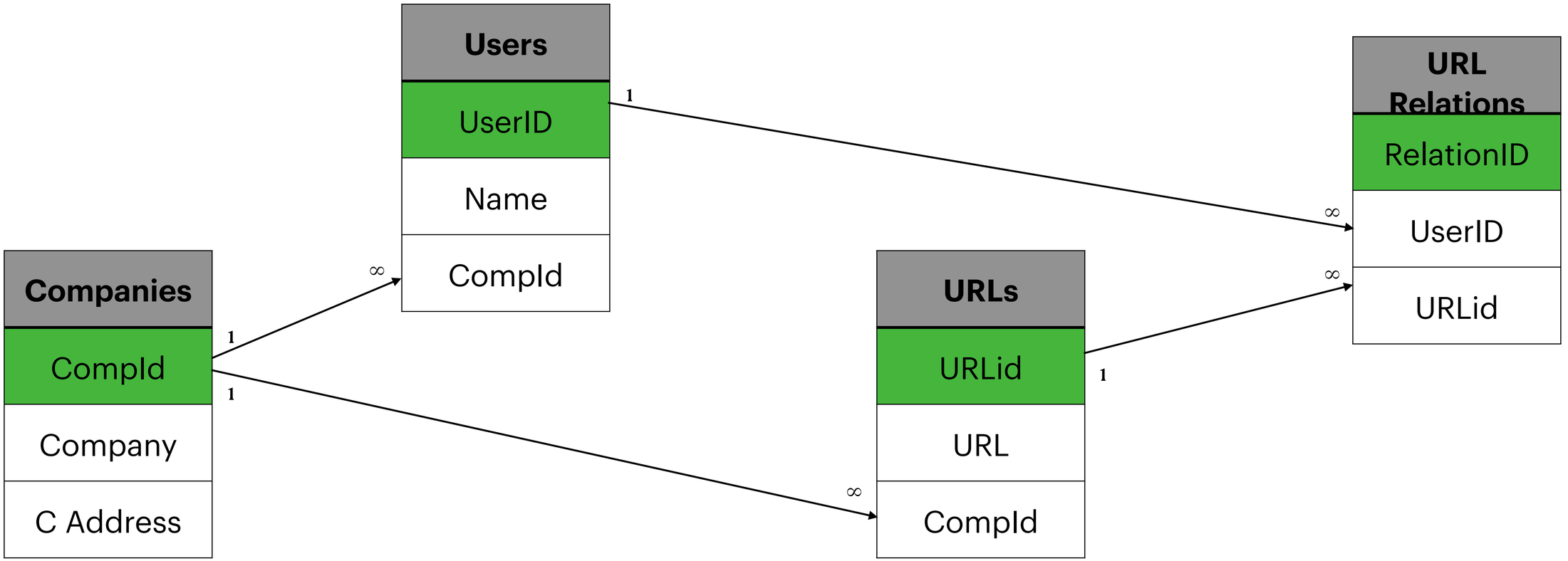
companies		
compId	company	company_address
1	ABC	1 Work Lane
2	XYZ	1 Job Street

urls	
urlId	url
1	abc.com
2	xyz.com

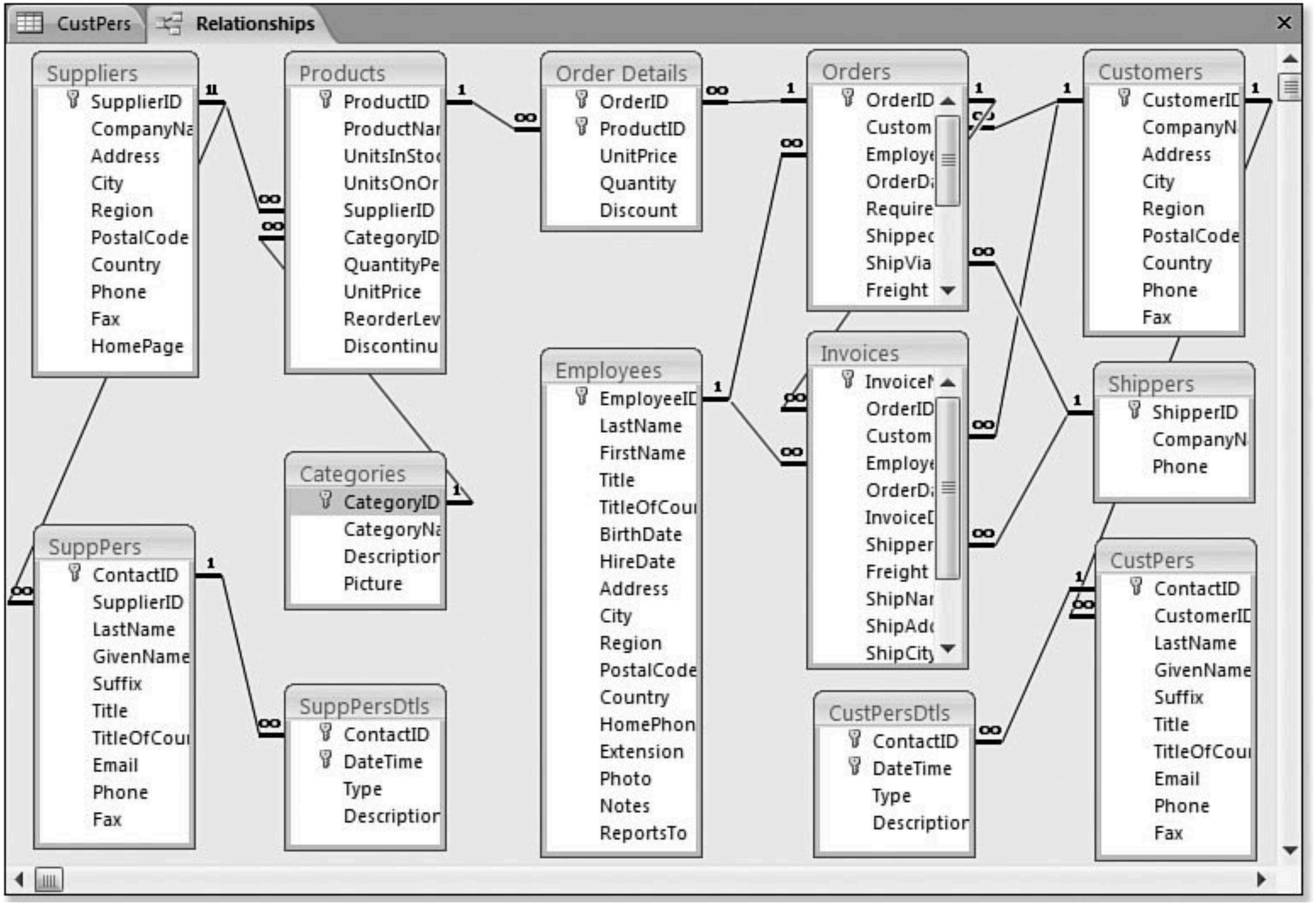
url_relations		
relationId	relatedUrlId	relatedUserId
1	1	1
2	1	2
3	2	1
4	2	2

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

Representing a database



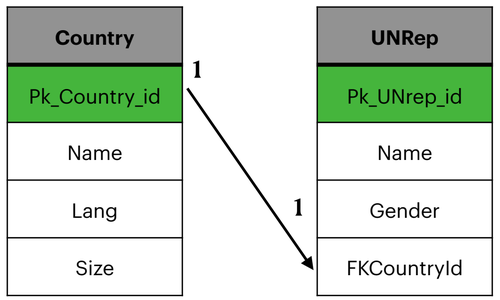
- Table Name at the top (grey background)
- Primary key next (green background)
- Arrows indicate foreign keys
- Numbers by arrows indicate type of relationship


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

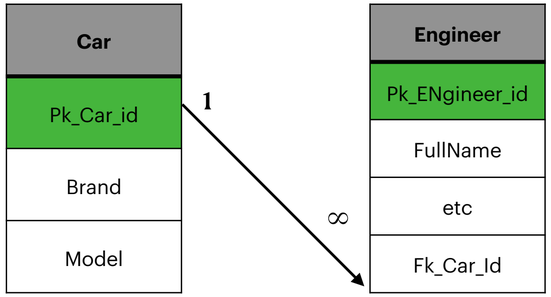
- One-to-One

```
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

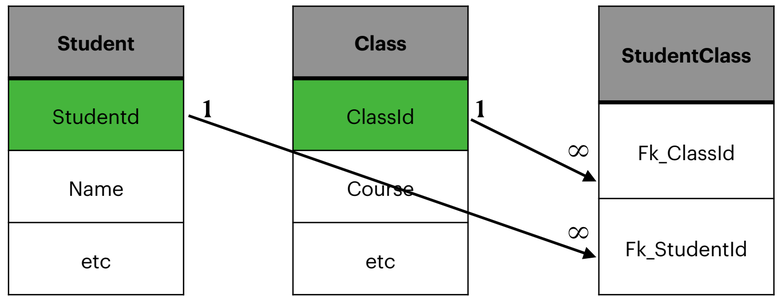
```
CREATE TABLE Car
(
Pk_Car_Id INT PRIMARY KEY,
Brand VARCHAR(100),
Model VARCHAR(100)
);

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)
);
```

Use 2 1-∞ relations to create a ∞-∞ relation



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 — CompanyID rather than just ID
- Use a single column for the primary key whenever possible; multi-column primary keys are appropriate for many-to-many relationships but rarely otherwise
- Use lookup tables rather than storing long values
  - Lookup tables are tables that have a key and a name … for instance: urls4, company4 table … They tend to be static
  - “lookup table” is in the eye of the beholder
- 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 URL1 and URL2 in userso table) 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
  - Often add a prefix to names to indicate FK relation — for instance “fk_” or “rel”
- 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).

Android File Layout

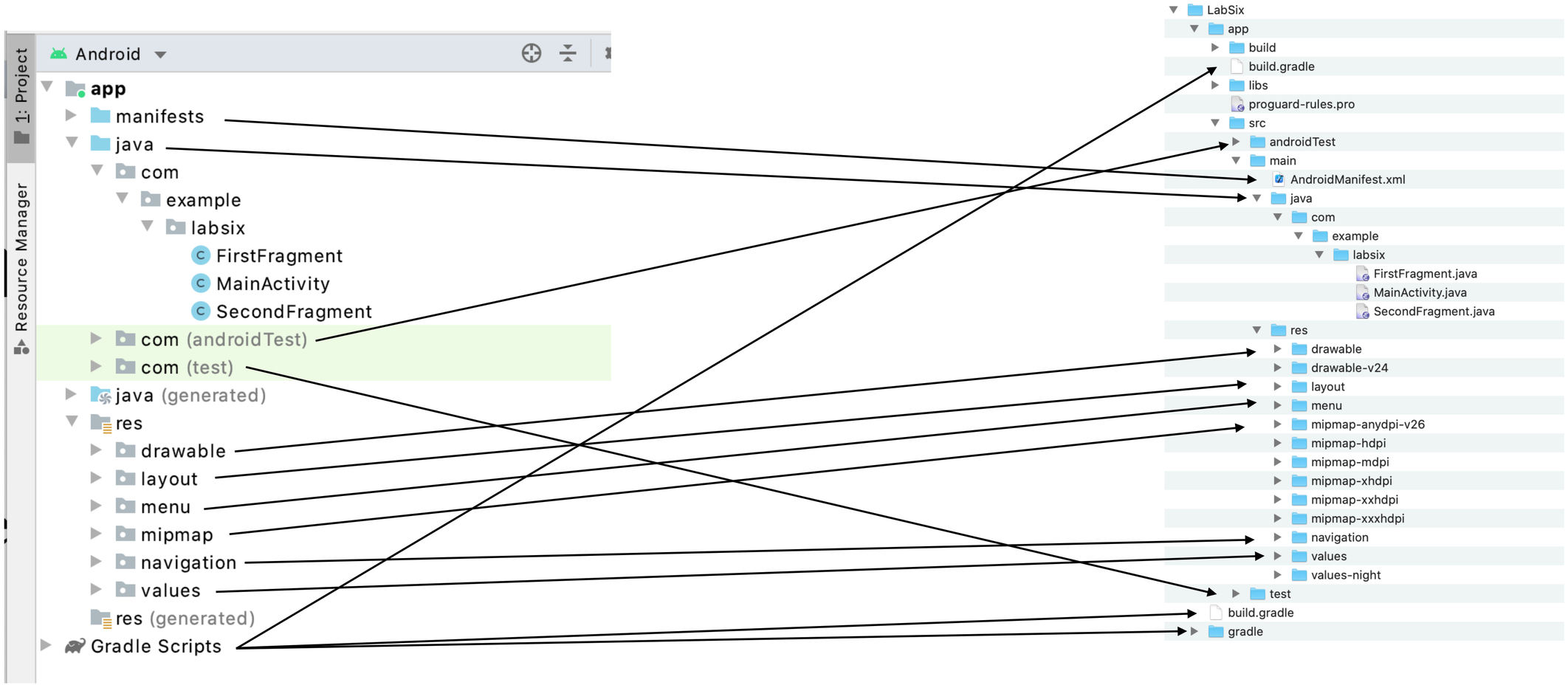
where things are and why they are there

- When you create a project, a directory is created that shares the name of your project
  - Typically in a folder name “AndroidStudioProjects” or something similar
- Everything important is in the “app” directory
  - Subdirs:
    - build — exactly what you think
    - libs — java jars and the like — often empty
    - src — everything you do is here
      - AndroidTest
        - test cases — testing code for android function (for discussion take Software Engineering in spring)
      - main
        - the actual running code
      - test
        - test cases - for other stuff

The src/main subdirectory

Almost everything you do is here!

- More importantly, the structure visible in Android Studio echoes this directory (plus some)



So what are all of these files????

- Manifest
  - Shows as a directory in AS but it it usually just one file, not a directory in file system
  - Usually just contains one file: AndroidManifest.xml
    - set the visible name and icon of your app
    - controls how other apps can interact with your app
      - compare lab6 with vb
- Gradle Scripts
  - Gradle is the system that AS uses to assemble and compile the app
    - There are two(!) build.gradle files.
      - This is useful in very big, multi-person projects. Otherwise just annoying
      - compare lab6 with vb
    - targetsdk, vs minsdk
  - Lots of other things
    - proguard — Java obfuscation — why?
- assets
  - folder not created by default but really useful.
    - put static data files used by your app in here. (e.g., json, csv, or xml data files)
      - Why not put these on a server and download (and then cache)

The res directory

Separating presentation and preparation

- drawable / drawable-v24
  - Images. 2 directories allow you to have identical images with different resolutions.
    - There uses to be a billion of these directories
- mipmap* — “multum in parvo” — “much in little”
  - app/launcher icons. Lots of directories to support lots of different sized devices.
    - You might make a very different image for 20x20 than you would for 40x40. The directories allow you to do so.
      - the default icon in mipmap-mdpi is 48x48.
      - In xxxhdpi it is 192x192
- Values
  - strings.xml
    - ideally, all user-facing strings are in this file
      - Internationalization!
      - AS: right-click on string in code to get a dialog to move it to strings.xml.
  - colors.xml, ...

More in res

- Layout
  - Design screens in app
    - Almost total separation presentation from preparation
      - In any but very small shops, there a people who never work outside of the res directory and its files — in particular the layout system
    - The separation is not perfect
      - The menu and the email button
- Menu:
  - organizes dropdown and slideout menus
    - demo add menu item to 3 dots

Measuring/positioning things

Android

- px (Pixels) -
  - Actual pixels or dots on the screen.
- in (Inches) -
  - Physical size of the screen in inches.
- mm (Millimeters) -
  - Physical size of the screen in millimeters.
- pt (Points)
  - 1/72 of an inch.
- dp (Density-independent Pixels) -
  - An abstract unit that is based on the physical density of the screen. These units are relative to a 160 dpi screen, so one dp is one pixel on a 160 dpi screen. The ratio of dp-to-pixel will change with the screen density, but not necessarily in direct proportion. "dip" and "dp" are same.
- sp (Scale-independent Pixels)
  - Similar to dp unit, but also scaled by the user's font size preference

HTML

- cm, mm, in
  - centimeters, millimeters, inches (1in = 96px = 2.54cm)
- px, pt, pc
  - pixels (1px = 1/96th of in), points (1pt = 1/72 of in), picas (1pc = 12 pt = 1/844 of in)
- em
  - Relative to the font-size of the element (2em means 2 times the size of the current font)
- ex
  - Relative to the x-height of the current font (rarely used)
- ch
  - Relative to the width of the "0" (zero)
- rem
  - Relative to font-size of the root element
- vw (vh)
  - Relative to 1% of the width, height of the viewport*
- vmin (vmax)
  - Relative to 1% of viewport's* smaller (larger) dimension
- %
  - Relative to the parent element