Multiple Tables

CMPSC 305 – Database Systems

ALLEGHENY COLLEGE

Importing Data: Two ideas

First idea: INSERT line-by-line

- Slow to code and to make the insertions

Populating tables line-by-line

```
INSERT INTO Table VALUES("1","ww","xx","yy","zz");
```

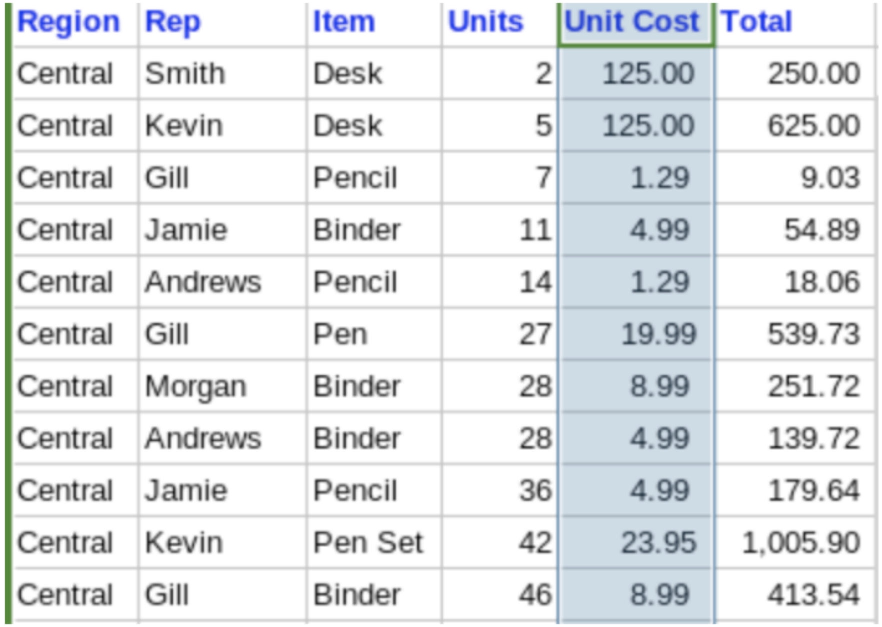
Second Idea: IMPORT CSV data files via a "build" file.

- Fast and you can easily rebuild your DB from files with updated data

Populating table-by-table

```
/*data files */
.import data/department.csv Department
```

Automate Data Entry
How to get data into a base?

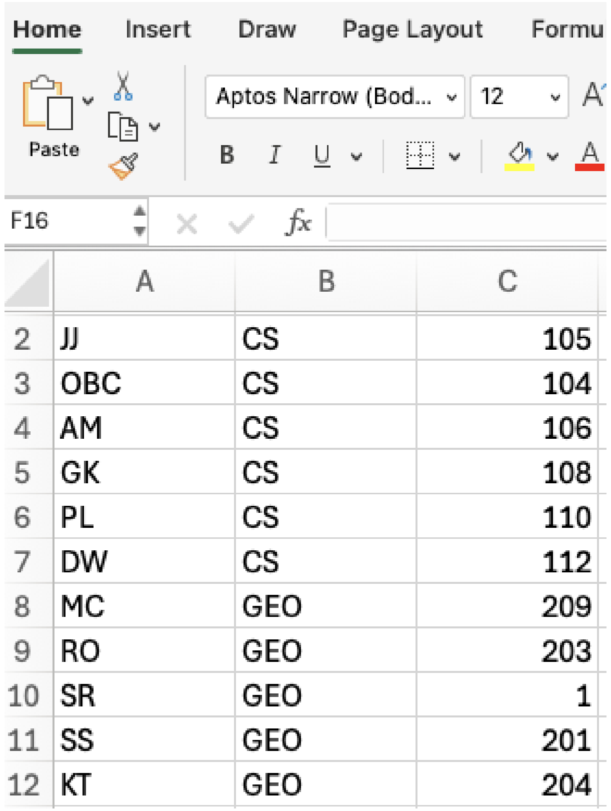


Region	Rep	Item	Units	Unit Cost	Total
Central	Smith	Desk	2	125.00	250.00
Central	Kevin	Desk	5	125.00	625.00
Central	Gill	Pencil	7	1.29	9.03
Central	Jamie	Binder	11	4.99	54.89
Central	Andrews	Pencil	14	1.29	18.06
Central	Gill	Pen	27	19.99	539.73
Central	Morgan	Binder	28	8.99	251.72
Central	Andrews	Binder	28	4.99	139.72
Central	Jamie	Pencil	36	4.99	179.64
Central	Kevin	Pen Set	42	23.95	1,005.90
Central	Gill	Binder	46	8.99	413.54

Data and CSV Files

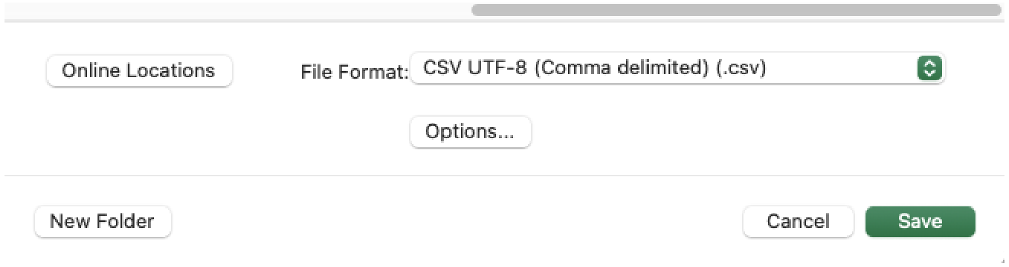
- Most data may arrive as a spreadsheet to be made into file
- Copy and Pasting is slow…

Data in Spreadsheet Form to Files



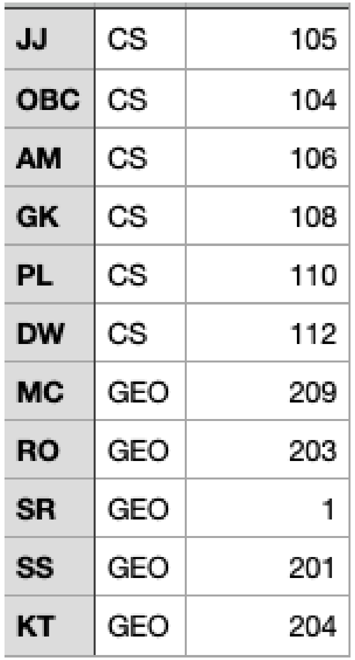
	A	B	C
2	JJ	CS	105
3	OBC	CS	104
4	AM	CS	106
5	GK	CS	108
6	PL	CS	110
7	DW	CS	112
8	MC	GEO	209
9	RO	GEO	203
10	SR	GEO	1
11	SS	GEO	201
12	KT	GEO	204

Use: "save as"

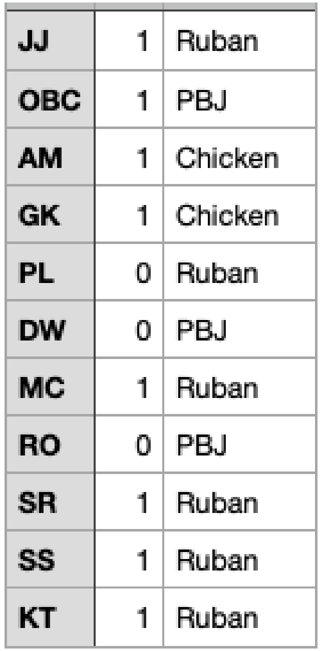


File Format: CSV UTF-8 (Comma delimited) (.csv)

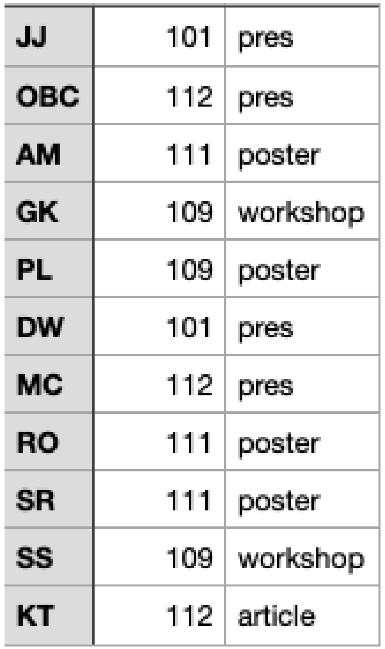
CSV Files



JJ	CS	105
OBC	CS	104
AM	CS	106
GK	CS	108
PL	CS	110
DW	CS	112
MC	GEO	209
RO	GEO	203
SR	GEO	1
SS	GEO	201
KT	GEO	204



JJ	1	Ruban
OBC	1	PBJ
AM	1	Chicken
GK	1	Chicken
PL	0	Ruban
DW	0	PBJ
MC	1	Ruban
RO	0	PBJ
SR	1	Ruban
SS	1	Ruban
KT	1	Ruban



JJ	101	pres
OBC	112	pres
AM	111	poster
GK	109	workshop
PL	109	poster
DW	101	pres
MC	112	pres
RO	111	poster
SR	111	poster
SS	109	workshop
KT	112	article

- Each table has own data file: Department, Tea, Session
- Be sure to use a text editor to remove any spaces and unusual characters!

Bring the data!

CREATE Tables: Build your create tables code before you load the data.
.separator: tells SQLite3 how the entries in the files are separated.
.import file.csv tableName: Pull data from and pour into a specific table.

Check sandbox for builder file

```
/*Table Creation Above in This File*/

.separator ",”

/*data files */
.import data/department.csv Department
.import data/tea.csv Tea
.import data/session.csv Session
```

Example

```
sqlite3 myteadb.sqlite3

DROP TABLE  IF EXISTS Tea;

CREATE TABLE Tea (
id TEXT NOT NULL PRIMARY KEY,
tea INTEGER NOT NULL,
sandwich  TEXT NOT NULL
);

.separator ",”

.import data/tea.csv Tea
```

Bring the data!

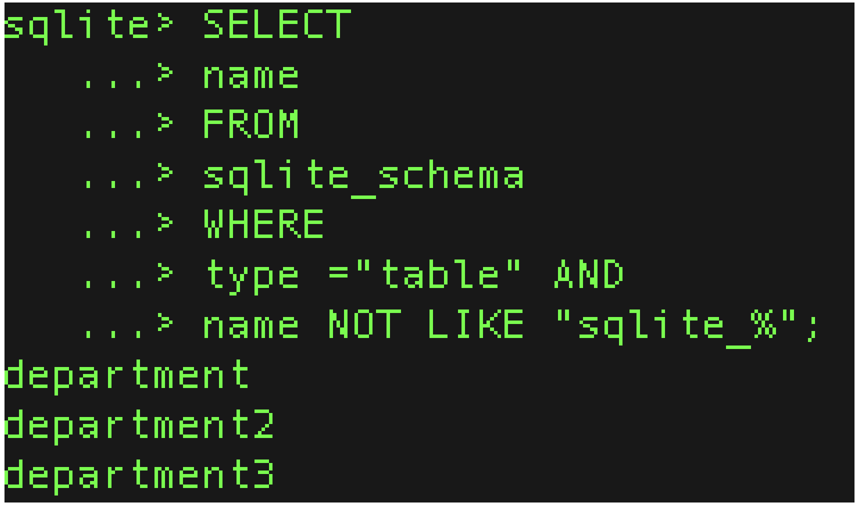
cat buildFile.txt | sqlite3 mydb.sqlite3

- UNIX: *cat buildFile.txt* : pull the contents of the text file (DB "builder file")
- " | " means to pipe the contents to next function in bash command
- *sqlite3 mydb.sqlite3* : Have sqlite3 create a file (called, mydb.sqlite3) to contain the database and data.
- Windows: Just copy and paste your SQL code from a builder file

New Way To List SQL Tables

List Tables command

```
SELECT
        name
FROM
        sqlite_schema
WHERE
        type ="table" AND
        name NOT LIKE "sqlite_%";
```



```
sqlite> SELECT
   ...> name
   ...> FROM
   ...> sqlite_schema
   ...> WHERE
   ...> type ="table" AND
   ...> name NOT LIKE "sqlite_%";
department
department2
department3
```

Importing Data: Two ideas

```
DROP TABLE IF EXISTS myTable1;
CREATE TABLE myTable1 (
        id INTEGER NOT NULL PRIMARY KEY,
        firstName VARCHAR NOT NULL);

DROP TABLE IF EXISTS myTable2;
CREATE TABLE myTable2 (
        id INTEGER NOT NULL PRIMARY KEY,
        firstName VARCHAR NOT NULL);
```

Then, the query (from above) to get the tables

```
SELECT name FROM sqlite_schema
WHERE type ="table" AND name NOT LIKE "sqlite_%";
```

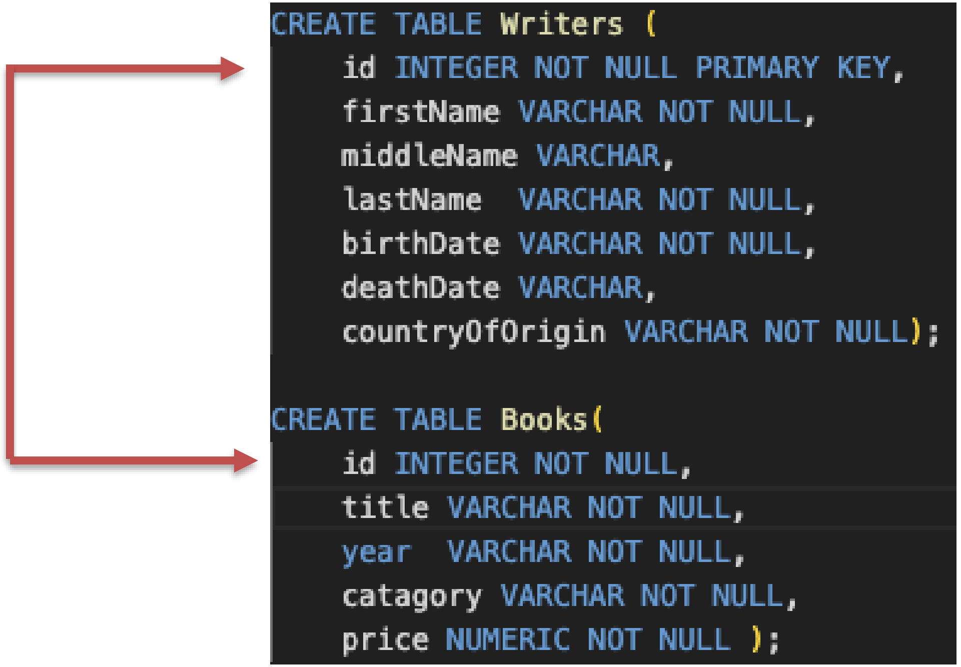
Overview of The Entity-Relationship Model Design

Consider these

- What is the data to store in the database?
- What are the relationships between the entities of information?
- What is the conceptual design of a system to link all this information together?

Table Relationship

- Relationship: the id is a primary key meaning that this attribute makes every row unique **AND** may be used to match authors to books
- Here, since both tables have unique id's relationship, we can link tables



```
CREATE TABLE Writers (
    id INTEGER NOT NULL PRIMARY KEY,
    firstName VARCHAR NOT NULL,
    middleName VARCHAR,
    lastName  VARCHAR NOT NULL,
    birthDate VARCHAR NOT NULL,
    deathDate VARCHAR,
    countryOfOrigin VARCHAR NOT NULL);

CREATE TABLE Books(
    id INTEGER NOT NULL,
    title VARCHAR NOT NULL,
    year  VARCHAR NOT NULL,
    catagory VARCHAR NOT NULL,
    price NUMERIC NOT NULL );
```

Create the file!

The terminal command to open a new database

```
sqlite3 writersAndBooksDB.sqlite3
```

The Writers Table

```
DROP TABLE IF EXISTS Writers;
CREATE TABLE Writers (
        id INTEGER NOT NULL PRIMARY KEY,
        firstName VARCHAR NOT NULL,
        middleName VARCHAR,
        lastName VARCHAR NOT NULL,
        birthDate VARCHAR NOT NULL,
        deathDate VARCHAR,
        countryOfOrigin VARCHAR NOT NULL);
```

- .schema Writers
- Note: id INTEGER NOT NULL PRIMARY KEY,
    - This attribute will be used to ensure all rows are unique
    - Used to connect to other tables

Adding Data to Writers Table

```
INSERT INTO Writers VALUES(1, "Francis","Scott",
        "Fitzgerald", "24Sept1896", "21Dec1940", "USA");

INSERT INTO Writers VALUES(2, "Arthur", "Conan",
        "Doyle", "22May1859", "7July1930", "UK");

INSERT INTO Writers VALUES(3, "Ernest", "Miller",
        "Hemingway", "21July1899", "2July1961", "USA");

INSERT INTO Writers VALUES(4, "John", "Edward",
        "Williams", "29Aug1922", "3Mar1994", "USA");
```

The Books Table

```
DROP TABLE IF EXISTS Books;
CREATE TABLE Books(
        id INTEGER NOT NULL,
        title VARCHAR NOT NULL,
        year VARCHAR NOT NULL,
        category VARCHAR NOT NULL,
        price NUMERIC NOT NULL);
```

- .schema Books
- All attributes must be present since "NOT NULL" is the rule

Add Data to the Books Table

```
/* Populate the Books table */

INSERT INTO Books VALUES(1," The Great Gatsby","1925","F",5);
INSERT INTO Books VALUES(1,"This Side of Paradise","1920","F",8);
INSERT INTO Books VALUES(1,"Tender is the Night","1934","F",9.50);
INSERT INTO Books VALUES(1," A Life in Letters","1975","nF",15);

/********************************************************************************/

INSERT INTO Books VALUES(2,"The Hound of the Baskervilles","1902","D",6.50);
INSERT INTO Books VALUES(2,"The Adventures of Sherlock Holmes","1892","D",10);
INSERT INTO Books VALUES(2,"The Lost World","1912","D",13);
INSERT INTO Books VALUES(2,"The Valley of Fear","1915","D",6);
```

Add Data to the Books Table

```
/* Populate the table */

INSERT INTO Books VALUES(3,"The Old Man and the Sea","1951","H",10);
INSERT INTO Books VALUES(3,"Men Without Women","1927","H",12);
INSERT INTO Books VALUES(3,"A Moveable Feast: The Restored Edition","2009","nH",15);
INSERT INTO Books VALUES(3,"Green Hills of Africa","1935","H",15);

/**********************************************************************************/

INSERT INTO Books VALUES(4,"Stoner","1965","W",27);
INSERT INTO Books VALUES(4,"Nothing but the Night","1948","W",14);
INSERT INTO Books VALUES(4,"Butcher’s Crossing","1960","W",20);
INSERT INTO Books VALUES(4,"The Broken Landscape: Poems","1949","W",20);
```

Queries!

Show me all rows in the Books table

- SELECT * FROM Books;

Show me all rows in the Writers table

- SELECT * FROM Writers;

Show me only rows for writers in the table who are from USA

- SELECT * FROM Writers WHERE countryOfOrigin == "USA";

Drawing from both tables simultaneously

Show me all books written by each of the writers.

```
SELECT
        Writers.lastName, Books.title
FROM
        Writers, Books
WHERE
        Writers.ID == Books.ID;

SELECT Writers.lastName, Books.title FROM Writers, Books
WHERE Writers.ID == Books.ID;
```

What do these queries give you?

```
SELECT Writers.lastName, Books.title
          FROM Writers, Books WHERE Writers.ID == Books.ID;

SELECT Writers.lastName, Books.title, Books.year
          FROM Writers, Books WHERE Writers.ID == Books.ID;

SELECT Writers.lastName, Books.title
          FROM Writers, Books WHERE writers.ID == Books.ID AND
          Writers.firstName == "Ernest";
```

What do these queries give you?

```
SELECT
        Writers.lastName, Writers.birthDate,
        Books.title, Books.year, Books.price
FROM
        Writers, Books
WHERE
        Writers.ID == Books.ID;

SELECT
        Writers.lastName, Books.category, Books.title, Books.year
FROM
        Writers, Books
WHERE Writers.ID == Books.ID;
```

What do these queries give you?

```
SELECT
        Writers.lastName, Writers.birthDate,
        Books.title, Books.year, Books.price
FROM
        Writers, Books
Where
        Writers.ID == Books.ID and price < 12;
```

Fix the below query

```
SELECT
          Writers.lastName, Books.title
FROM
          Writers, Books
WHERE
          Writers.ID == Writers.ID AND category == "D";
```

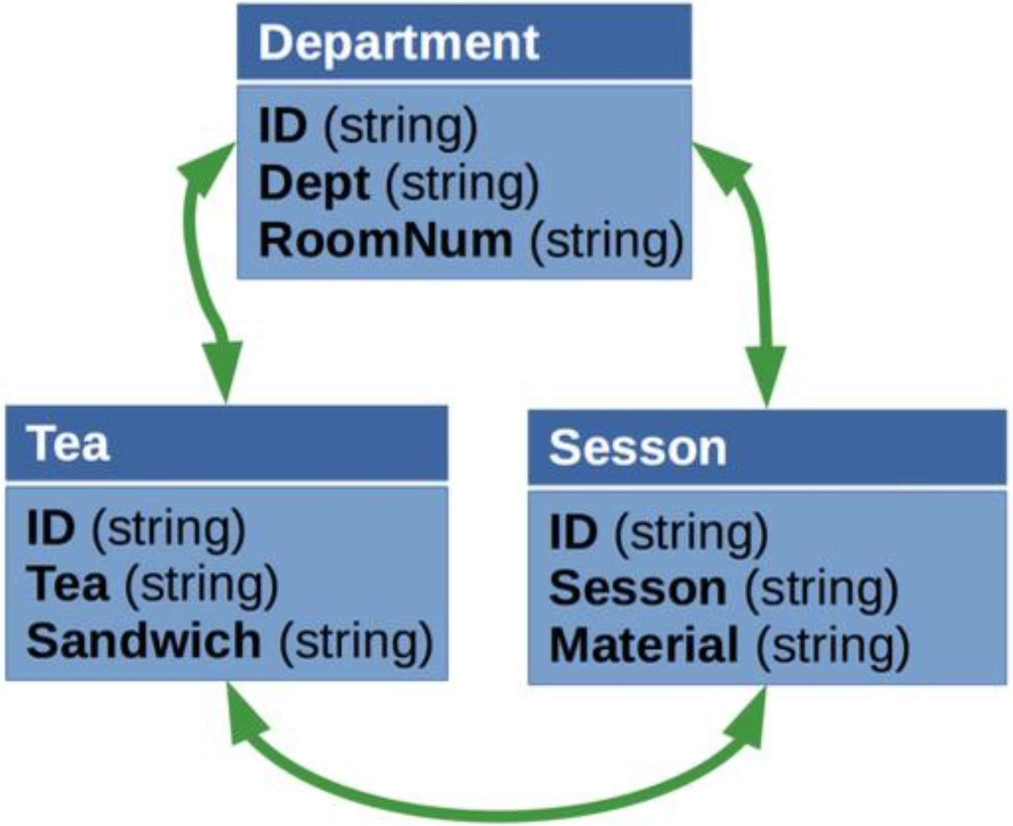
Output

```
Doyle|The Hound of the Baskervilles
Doyle|The Adventures of Sherlock Holmes
Doyle|The Lost World
Doyle|The Valley of Fear
```

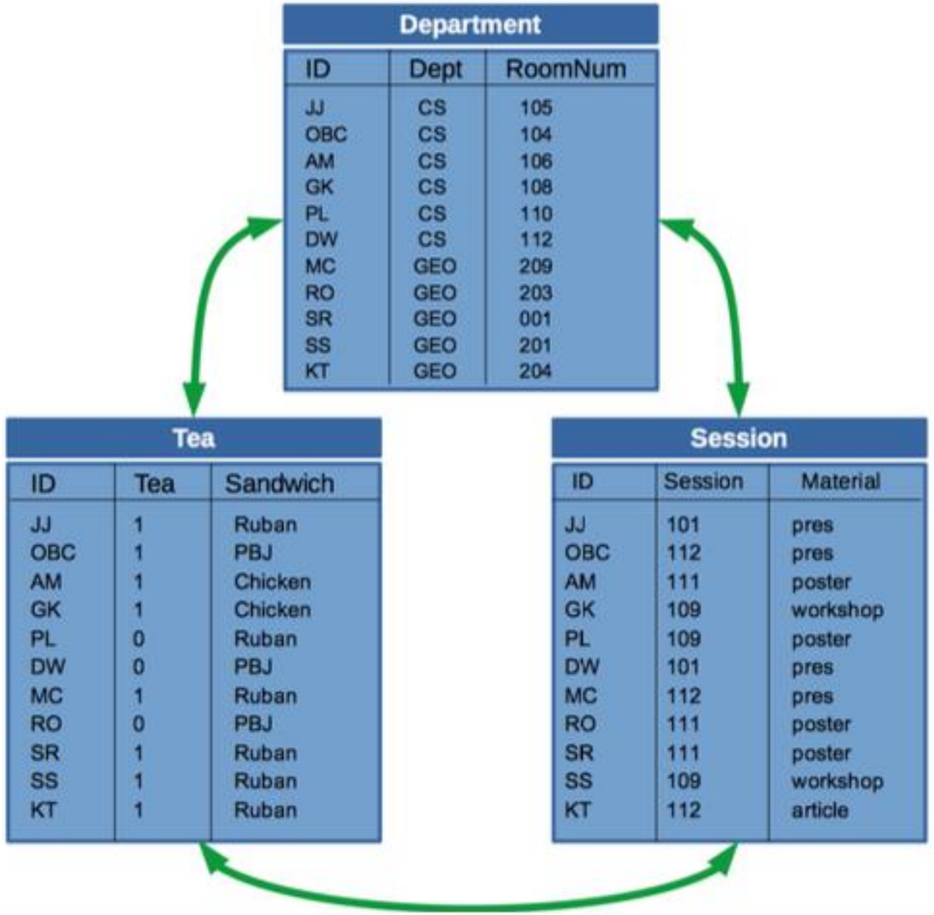
Importing Data: Two ideas

- Can you design and populate a database?
- Can you run queries to access particular attributes?

Connecting the Tables - Basic Schema



Find relationship to link



Department

ID	Dept	RoomNum
JJ	CS	105
OBC	CS	104
AM	CS	106
GK	CS	108
PL	CS	110
DW	CS	112
MC	GEO	209
RO	GEO	203
SR	GEO	001
SS	GEO	201
KT	GEO	204

Tea

ID	Tea	Sandwich
JJ	1	Ruban
OBC	1	PBJ
AM	1	Chicken
GK	1	Chicken
PL	0	Ruban
DW	0	PBJ
MC	1	Ruban
RO	0	PBJ
SR	1	Ruban
SS	1	Ruban
KT	1	Ruban

Session

ID	Session	Material
JJ	101	pres
OBC	112	pres
AM	111	poster
GK	109	workshop
PL	109	poster
DW	101	pres
MC	112	pres
RO	111	poster
SR	111	poster
SS	109	workshop
KT	112	article

Example Code Department Table

```
DROP TABLE IF EXISTS Department;
CREATE TABLE Department (
id TEXT NOT NULL PRIMARY KEY,
dept TEXT NOT NULL,
roomNum INTEGER NOT NULL
);
```

Populate

```
INSERT INTO Department Values ("JJ","CS",102);
INSERT INTO Department Values ("OBC","CS",203);
INSERT INTO Department Values ("DL","CS",106);
INSERT INTO Department Values ("HZ","CS",105);
```

Example Code Tea Table

```
DROP TABLE IF EXISTS Tea;
CREATE TABLE Tea (
id TEXT NOT NULL PRIMARY KEY,
tea INTEGER NOT NULL,
sandwich TEXT NOT NULL
);
```

Populate

```
INSERT INTO Tea Values ("JJ",1,"Ruban");
INSERT INTO Tea Values ("OBC",1,"Salad");
INSERT INTO Tea Values ("DL",0,"Chicken");
INSERT INTO Tea Values ("HZ",0,"Pizza");
```

Example Code Session Table

```
DROP TABLE IF EXISTS Session;
CREATE TABLE Session (
id TEXT NOT NULL PRIMARY KEY,
session INTEGER NOT NULL,
material TEXT NOT NULL
);
```

Populate

```
INSERT INTO Session Values ("JJ ", 101, "pres");
INSERT INTO Session Values ("OBC", 112, "pres");
INSERT INTO Session Values ("DL", 119, "poster");
INSERT INTO Session Values ("HZ", 103, "poster");
```

Queries

Single table, Show me all rows from each of the tables, individually.

Two tables, Show me the name, dept and whether the person will have tea.

Two tables, Show me the name and dept of each person who will have a Ruban.

Two tables, Show me the sandwich type and the session number of each person.

Show the ID, sandwich type, session material and department room number.

Queries

Three table3, Show the ID, sandwich type, session material and department room number.

Three table3, Show me all the ID, Material, Tea and Sandwich and department room number.

Three table3, Show me the ID, Material, Tea and Sandwich and department room number for each person who will have a Ruban.

Three table3, Show me the ID, Material, Tea and Sandwich and department room number for each person who will be presenting a “pres”.

THINK

- Can you design and populate a database of three tables?
- Can you run queries to access attribute(s) in three tables?