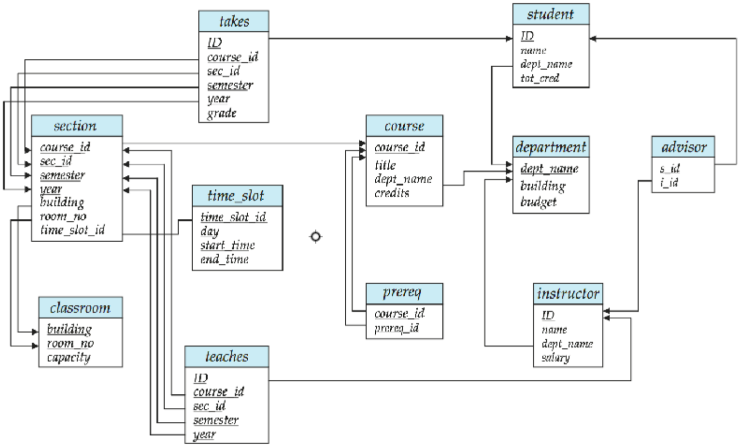
Introduction to Database Systems: CS312 SQL Queries, SELECT and WHERE

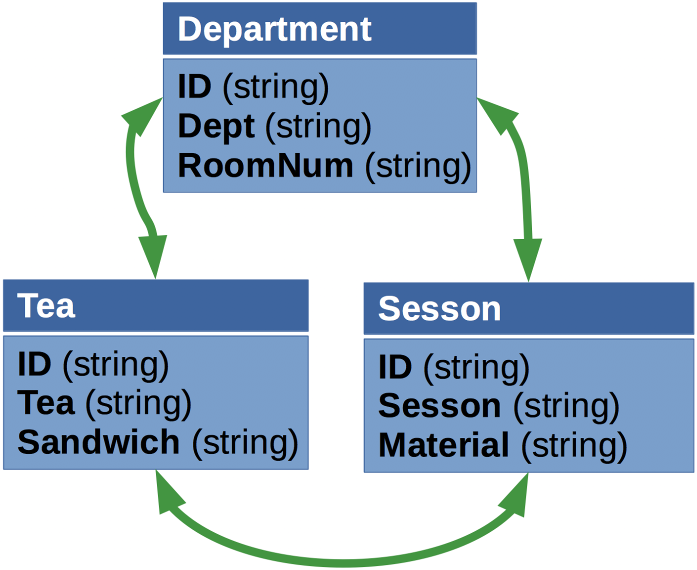
Oliver Bonham-Carter

1 February 2019

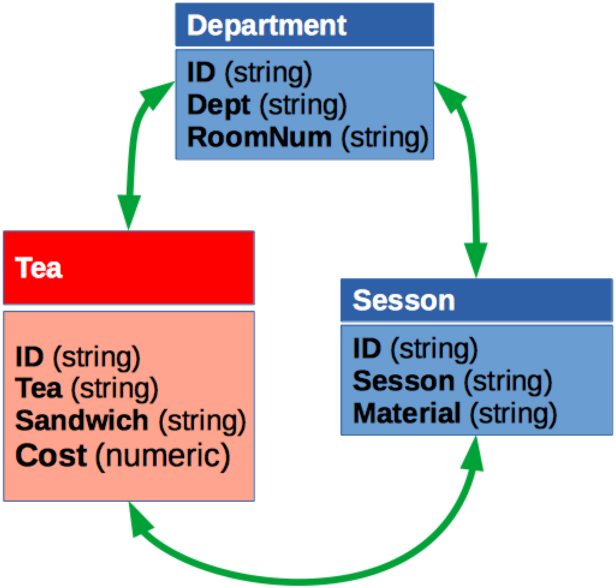
How to connect all this information?



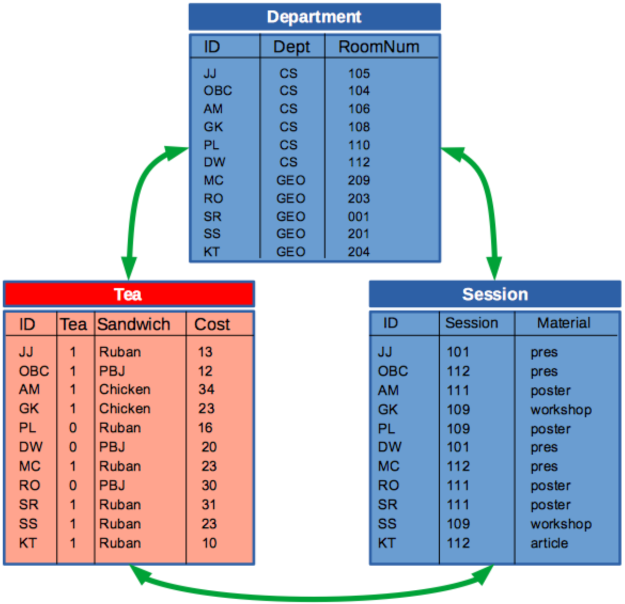
Previous table



New Table


Department
ID (string)
Dept (string)
RoomNum (string)
Tea
ID (string)
Tea (string)
Sandwich (string)
Cost (numeric)
Sesson
ID (string)
Sesson (string)
Material (string)

New Data in the Tea Table



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

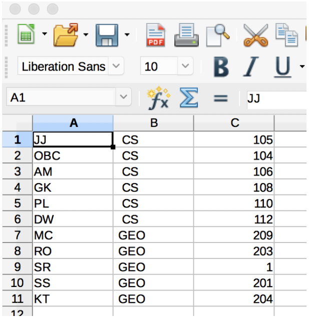
ID	Tea	Sandwich	Cost
JJ	1	Ruban	13
OBC	1	PBJ	12
AM	1	Chicken	34
GK	1	Chicken	23
PL	0	Ruban	16
DW	0	PBJ	20
MC	1	Ruban	23
RO	0	PBJ	30
SR	1	Ruban	31
SS	1	Ruban	23
KT	1	Ruban	10

Session

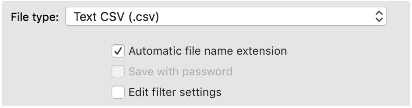
ID	Session	Material
JJ	101	pres
OBC	112	pres
AM	111	poster
GK	109	workshop
PL	109	poster
DW	101	pres
MC	112	pres
RO	111	poster
SR	111	poster
SS	109	workshop
KT	112	article

Data in Spreadsheet Form to Files

Remember to use: "save a copy"



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

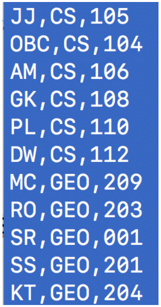


File type: Text CSV (.csv)

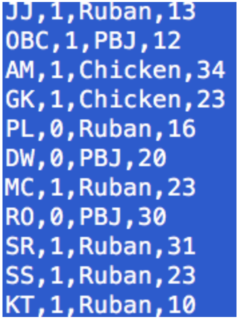
- [✓] Automatic file name extension
- [ ] Save with password
- [ ] Edit filter settings

- Remember from last class how to save a CSV file and to use an editor to remove the spaces
- Store your data files in the `data/`

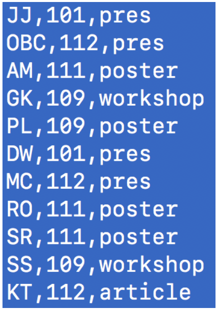
Making files of CSV's (Comma-separated values)...?



```
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
```



```
JJ,1,Ruban,13
OBC,1,PBJ,12
AM,1,Chicken,34
GK,1,Chicken,23
PL,0,Ruban,16
DW,0,PBJ,20
MC,1,Ruban,23
RO,0,PBJ,30
SR,1,Ruban,31
SS,1,Ruban,23
KT,1,Ruban,10
```



```
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
```

- Tables: *department*, *tea*, *session*
- We have added a numeric column (cost) to *Tea*

Department Table

```
DROP TABLE Department;
CREATE TABLE Department (
    id VARCHAR NOT NULL PRIMARY KEY,
    dept VARCHAR NOT NULL,
    roomNum VARCHAR NOT NULL
);
```

Tea Table

```
DROP TABLE Tea;
CREATE TABLE Tea (
     id VARCHAR NOT NULL PRIMARY KEY,
     tea VARCHAR NOT NULL,
     sandwich VARCHAR NOT NULL,
     cost numeric NOT NULL
);
```

- Note: New *numeric* attribute to the table: `cost`

Session Table

```
DROP TABLE Session;
CREATE TABLE Session (
    id VARCHAR NOT NULL PRIMARY KEY,
    session VARCHAR NOT NULL,
    material VARCHAR NOT NULL
);
```

- Find your sandbox database file with its updated data to *compile* your database

The Basic Query Structure

The SQL data-manipulation language (DML) provides the ability to query information, and insert, delete and update tuples

A typical SQL pseudo code query has the form:

```
select A1, A2, ..., An
from r1, r2, ..., rm
where P;
```

- A_i represents an attribute
- R_i represents a relation
- P is a predicate
- The result of an SQL query is a relation

The **select** Clause

The SELECT clause filters out particular data from a table.

- SQL allows duplicates in relations as well as in query results.
- The SELECT statement has many optional clauses:
    - WHERE specifies which rows to retrieve.
    - GROUP BY groups rows sharing a property so that an aggregate function can be applied to each group.
    - HAVING selects among the groups defined by the GROUP BY clause.
    - ORDER BY specifies an order in which to return the rows.
    - AS provides an alias which can be used to temporarily rename tables or columns..

Given table 'T'

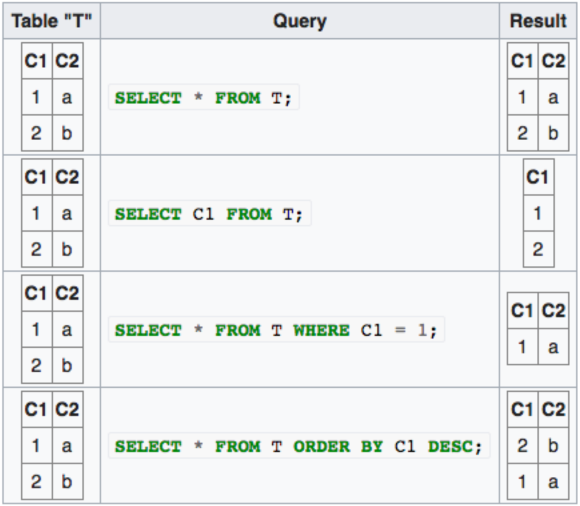


Table "T"	Query	Result
C1 C2 1 a 2 b	`SELECT * FROM T;`	C1 C2 1 a 2 b
C1 C2 1 a 2 b	`SELECT C1 FROM T;`	C1 1 2
C1 C2 1 a 2 b	`SELECT * FROM T WHERE C1 = 1;`	C1 C2 1 a
C1 C2 1 a 2 b	`SELECT * FROM T ORDER BY C1 DESC;`	C1 C2 2 b 1 a

The **select** Clause

- Find the names of all *depts* of the department table and remove all duplicates
  - SELECT DISTINCT(dept) FROM department; /*return a number*/
  - SELECT COUNT(DISTINCT(dept)) FROM department; /*count unique occurrences*/
- Return query for all *roomNum* of the department table and remove all duplicates
  - SELECT DISTINCT(roomNum) FROM Department;

The **where** clause

The where clause: conditions that the result must satisfy

- Corresponds to the selection predicate of the relational algebra
- Comparison results can be combined using the logical connectives and, or, and not
- Comparisons can be applied to results of arithmetic expressions

The **where** clause

- Find out who is ordering a sandwich less than $15
  - SELECT * FROM tea where cost $<$ 15;
- Find out what kinds of *sandwich*es are going to each *dept*
  - SELECT department.dept, tea.sandwich FROM department, tea where department.id $==$ tea.id;

The **where** clause

- Find out which professors are presenting posters
  - SELECT * FROM session WHERE material == "poster"; /* show all*/
  - SELECT ID, material FROM session WHERE material == "poster"; /*which professor is doing what?*/
- Find how who is presenting a poster, having what kind of sandwich which costs over $10
  - SELECT session.ID, session.material, tea.sandwich, tea.cost FROM session, tea WHERE session.material == "poster" AND tea.cost > 10 AND session.id == tea.id;

Consider this ...

- Can you run queries to solve the challenge on the next slide?

Query each table

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.

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

Three tables

Show me the sandwich type and the session room number of each person.

Can you think of other interesting queries here?