CS61A Discussion 11:
SQL

TA: **Jerry Chen**
Email: **jerry.c@berkeley.edu**
TA Website: **jerryjrchen.com/cs61a**

Attendance

Form: **tinyurl.com/jerrydisc**

For the weekly question, **list the topics you are most worried about for the final**

(Of course, please only check in if you showed up!)

Agenda

1. Week in Review
2. SQL Tables & Queries
3. Joins
4. Aggregation
5. Recursive Queries

Week In Review

Lab 12 (SQL) - **Due Friday**

Proj4 - **Due Monday**

- Submit a day early for an extra point

Ants Composition - Resubmit by 4/29

Hw8 is out! - Due 4/27

Databases

Data — information about pretty much anything

A **database** is an ordered collection of data

Use **tables** to organize data

Databases show up everywhere!



SQL

Structured Query Language

(Pronounced "Ess Cue El" or "Sequel")

Used to manage data stored in a database

A **declarative** language — broadly speaking, tell it **what we want**, not how to do it

All "queries" (expressions) end in a semicolon ";"

SQL

The **select** statement create tables

- Use the **union** command to join two select rows

The **create table** expression saves a table for later

Select

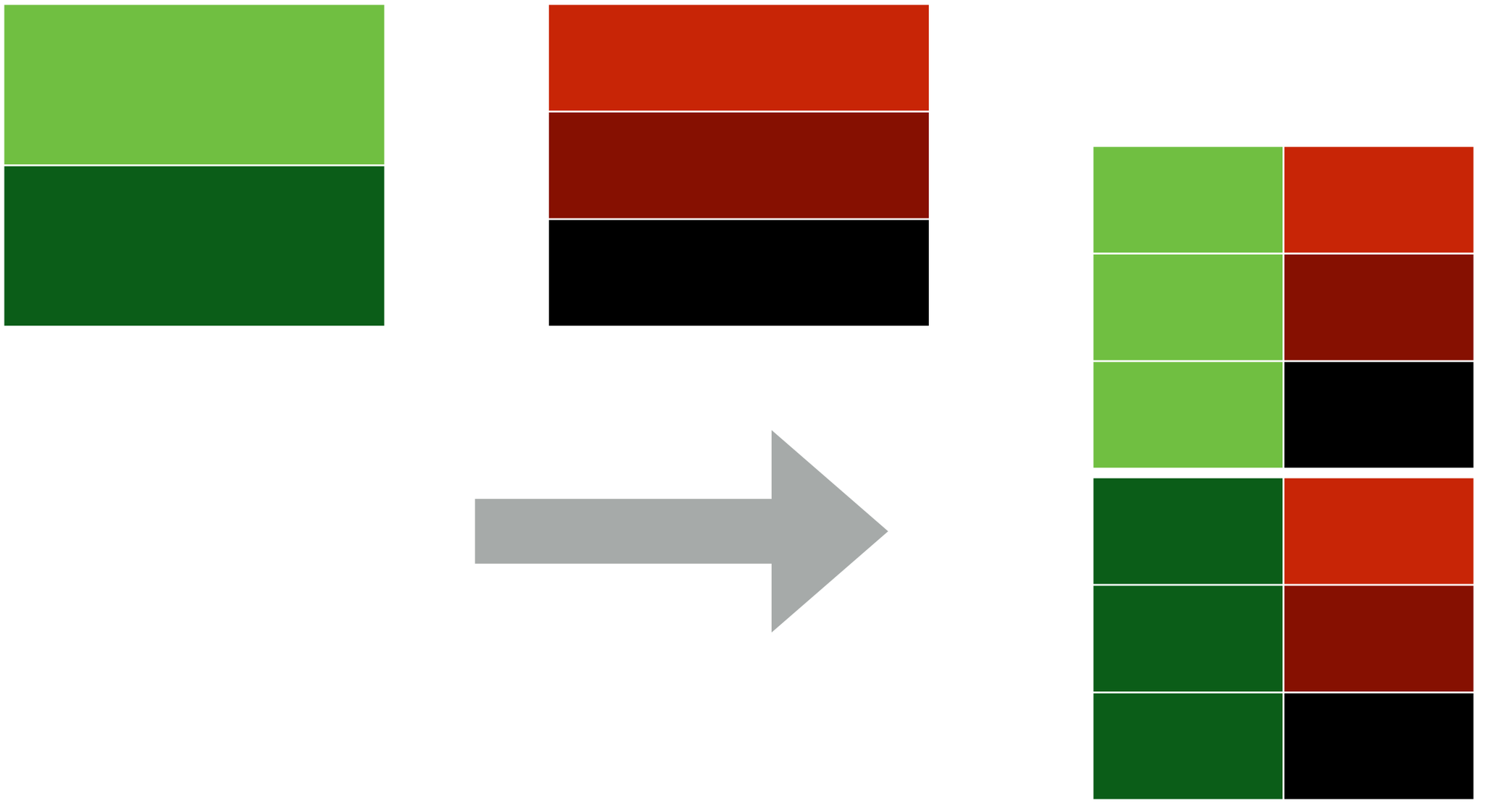
select doesn't have to start from scratch: can select **from** an existing table to create a new one

Specify what columns to keep in your result!

Filter results using boolean expressions in the **where clause**

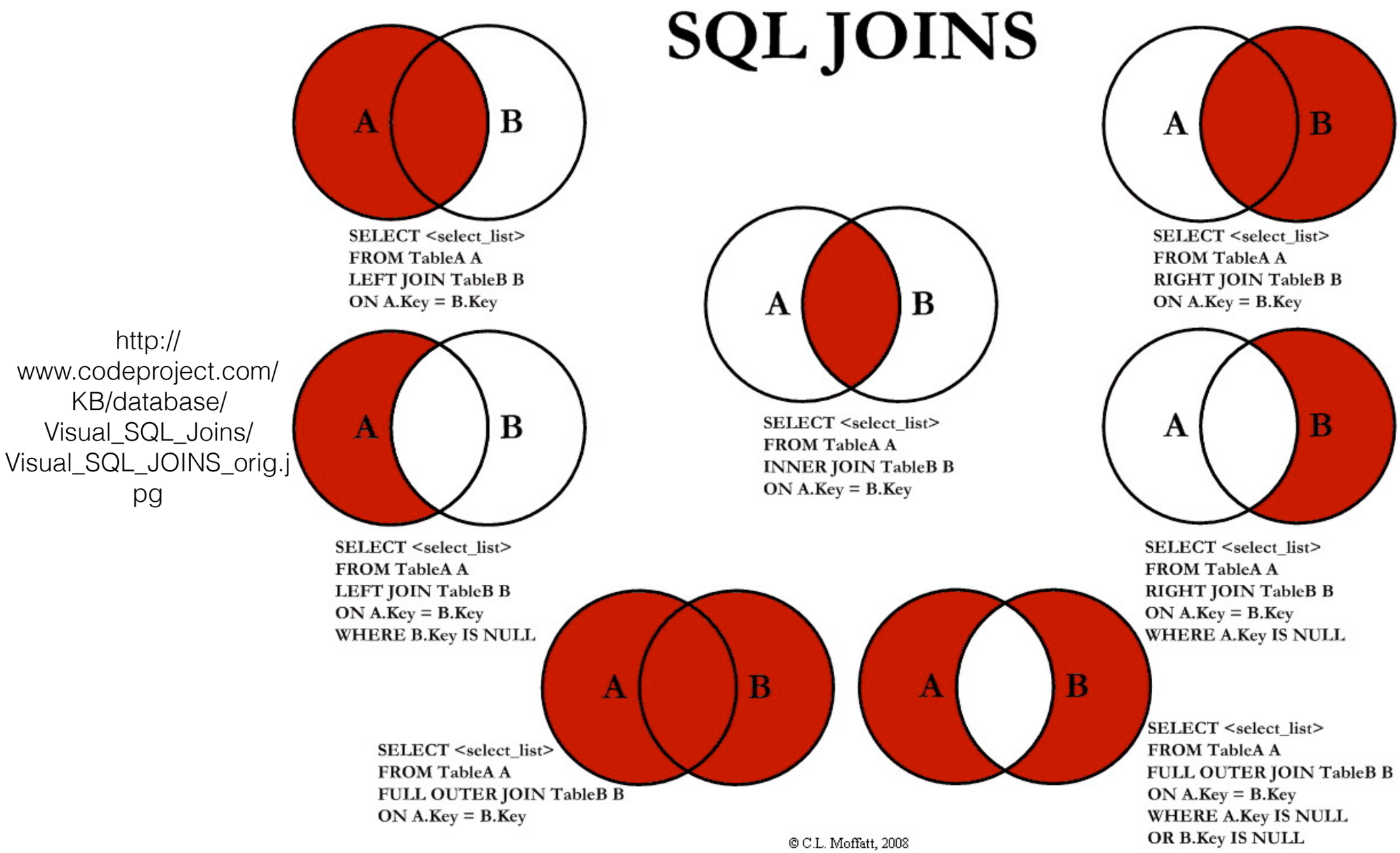
Joins

When we join two tables together, consider all possible pairings:



Joins

Of course, it gets more complicated (out of scope)



http://www.codeproject.com/KB/database/Visual_SQL_Joins/Visual_SQL_JOINS_orig.jpg

Joins

If we're joining with ourself (or a table with the same column names), we may require **aliasing**

Be wary of **duplicates** and **self-joins** (row joined to itself)!

- Solve by **enforcing ordering**

Aggregation

Big idea: examine **groups of rows** that have a trait in common

One possible group of course is **the entire table!**

Aggregate functions operate specifically on groupings

- Ex: Instead of adding two joined rows together, add all the rows in a group together

Aggregation

The **group by** command can be used to specify groupings by value

Aggregate functions will then operate on those groupings

Special **aggregate filtering** using **having** instead of where (filter entire groups instead of single rows)

Recursive Select

Like regular recursion start off with a **base row** (base case)

Subsequent rows based off of previous rules (recursive step)

Use filter (where) to determine when to stop

We usually use a local **with** table to create recursive tables

Closing Note

There's a lot more to databases and SQL than what we teach you in 61A (check out CS 186!)

In my opinion, they have become more and more important as we collect more and more data

Side note (out of scope): SQL injection attacks!