Tabular Data Model

+

Relational Databases

Outline

- What is a relational Database? What Grammar of Data does it follow?
- How is this grammar implemented in Pandas?
- How is this grammar implemented in SQL

Relational Database

- A collection of tables related to each other through common data values.
- Rows represent attributes of something
- Everything in a column is values of *one* attributes
- A cell is expected to be atomic
- Tables are related to each other if they have columns called keys which represent the same values

Table: contributors

New Record | Delete Record

Contributors

	id	last_name	first_name	middle_name	street_1	street_2	city	state	zip	amount	date	candidate_id
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	Agee	Steven	NULL	549 Laurel ...	NULL	Floyd	VA	24091	500	2007-06-30	16
2	5	Akin	Charles	NULL	10187 Sega...	NULL	Bentonville	AR	72712	100	2007-06-16	16
3	6	Akin	Mike	NULL	181 Baywo...	NULL	Monticello	AR	71655	1500	2007-05-18	16
4	7	Akin	Rebecca	NULL	181 Baywo...	NULL	Monticello	AR	71655	500	2007-05-18	16
5	8	Aldridge	Brittni	NULL	808 Capitol...	NULL	Washington	DC	20024	250	2007-06-06	16
6	9	Allen	John D.	NULL	1052 Cann...	NULL	North Augu...	SC	29860	1000	2007-06-11	16
7	10	Allen	John D.	NULL	1052 Cann...	NULL	North Augu...	SC	29860	1300	2007-06-29	16
8	11	Allison	John W.	NULL	P.O. Box 10...	NULL	Conway	AR	72033	1000	2007-05-18	16
9	12	Allison	Rebecca	NULL	3206 Sum...	NULL	Little Rock	AR	72227	1000	2007-04-25	16

	id	first_name	last_name	middle_name	party
	Filter	Filter	Filter	Filter	Filter
1	16	Mike	Huckabee		R
2	20	Barack	Obama		D
3	22	Rudolph	Giuliani		R
4	24	Mike	Gravel		D
5	26	John	Edwards		D
6	29	Bill	Richardson		D
7	30	Duncan	Hunter		R
8	31	Dennis	Kucinich		D
9	32	Ron	Paul		R

Candidates

Grammar of Data

Been there for a while (SQL, Pandas), formalized in `dplyr`[1].

- provide simple verbs for simple things. These are functions corresponding to common data manipulation tasks
- second idea is that backend does not matter. Here we constrain ourselves to Pandas and sqlite
- multiple backends implemented in Pandas, Spark, Impala, Pig, dplyr, ibis, blaze

[1] Hadley Wickham: https://cran.rstudio.com/web/packages/dplyr/vignettes/introduction.html

Why bother with SQL and the grammar

- learn hot to do core data manipulations, no matter what the system
- relational databases critical for mon-memory fits. Big installed base.
- dbs like Cloudera, Azure Data Lake implement SQL over clusters, multiple databases
- one off questions: google, stack-overflow, http://chrisalbon.com

VERB	dplyr	pandas	SQL
QUERY/SELECTION	`filter()` (and `slice()`)	`query()` (and `loc[]`, `iloc[]`)	`SELECT WHERE`
SORT	`arrange()`	`sort_values()`	`ORDER BY`
SELECT-COLUMNS/ PROJECTION	`select()` (and `rename()`)	`[](__getitem__)` (and `rename()`)	`SELECT` COLUMN
SELECT-DISTINCT	`distinct()`	`unique()`,`drop_duplicates()`	`SELECT DISTINCT` COLUMN
ASSIGN	`mutate()` (and `transmute()`)	`assign`	`ALTER/UPDATE`
AGGREGATE	`summarise()`	`describe()`, `mean()`, `max()`	None, `AVG()`,`MAX()`
SAMPLE	`sample_n()` and `sample_frac()`	`sample()`	implementation dep, use `RAND()`
GROUP-AGG	`group_by/summarize`	`groupby/agg`, `count`, `mean`	`GROUP BY`
DELETE	?	drop/masking	`DELETE/WHERE`

Operation	Pandas	SQL
QUERY	`dfcwci.query("state=='VA' & amount < 400")`	`SELECT * FROM contributors WHERE state='VA' AND amount < 400;`
SORT	`dfcwci.sort_values("amount")`	`SELECT * FROM contributors ORDER BY amount;`
SELECT-COLUMNS	`dfcwci[['first_name', 'amount']]`	`SELECT first_name, amount FROM contributors;`
SELECT-DISTINCT	`dfcwci[['last_name','first_name']].drop_duplicates()`	`SELECT DISTINCT last_name, first_name FROM contributors;`
ASSIGN	`dfcwci['name']=dfcwci['last_name']+", "+dfcwci['first_name']`	`ALTER TABLE contributors ADD COLUMN name; UPDATE contributors SET name = ? WHERE id = ?;`
AGGREGATE	`dfcwci.amount.max()`	`SELECT MAX(amount) FROM contributors;`
GROUP-AGG	`dfcwci.groupby("state").sum()`	`SELECT state,SUM(amount) FROM contributors GROUP BY state;`
DELETE	`dfcwci=dfcwci[dfcwci.last_name!='Ahrens']`	`DELETE FROM contributors WHERE last_name=\"Ahrens\";`
DELETE TABLE	`del dfcwci`	`DELETE FROM contributors;`
CREATE TABLE	`dfcwci=pd.read*csv("data/contributors*with*candidate*id.txt", sep="`	`")`

Split-Apply-Combine (GROUP-AGG)

1. split the data into groups
2. based on some criteria apply a function to each group independently
3. combinine the results into a data structure

```
dfcwci.groupby("state")['amount'].mean()
```

```
state
AK      403.333333
AR     1183.333333
AZ      120.000000
CA     -217.988261
CO    -1455.750000
CT     2300.000000
DC     -309.982000
...
```

RELATIONSHIPS

- we usually need to combine data from multiple sources
- different systems have diffferent ways, but most copy SQL
- sub-select:

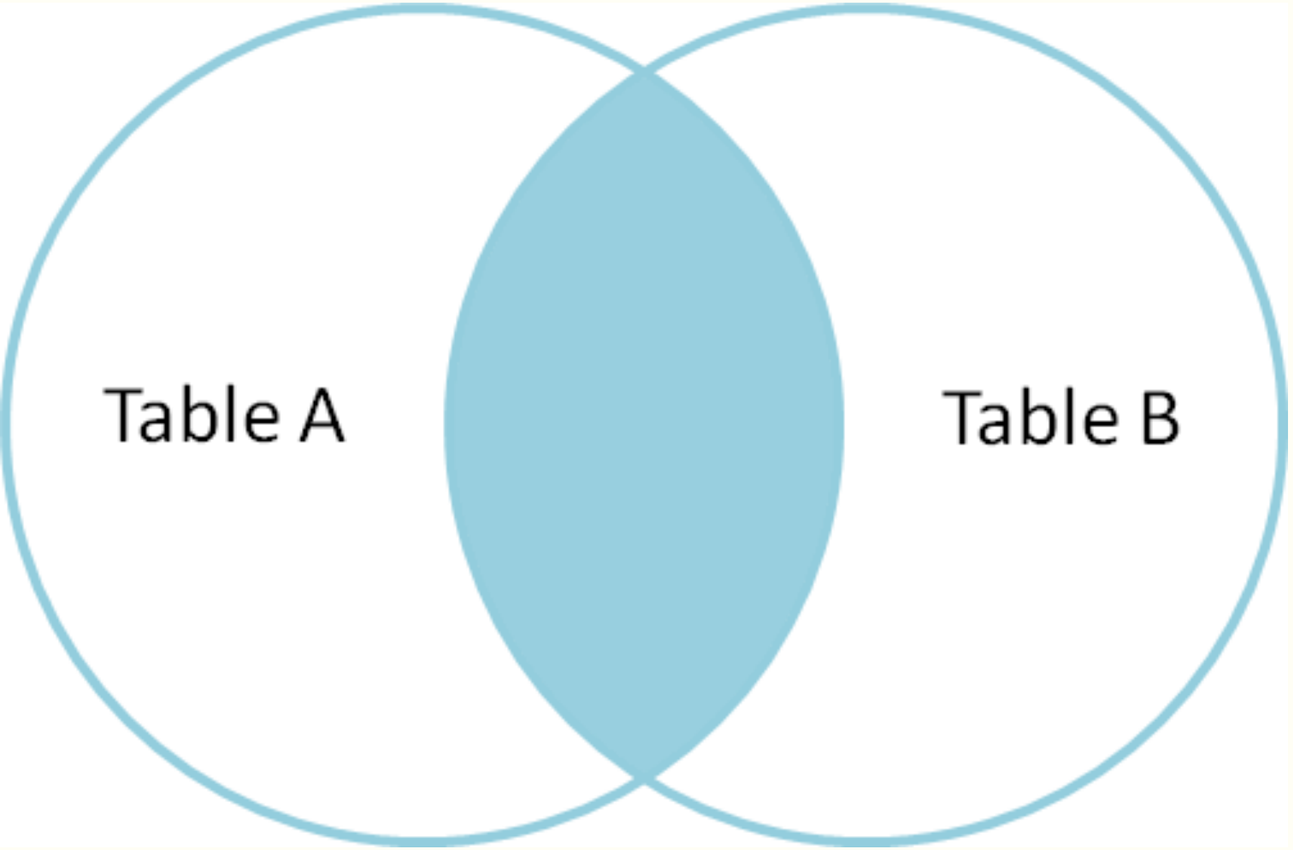
```
obamaid=dfcand.query("last_name=='Obama'")['id'].values[0]
dfcwci.query("candidate_id==%i" % obamaid)
```

```
SELECT * FROM contributors WHERE
    candidate_id = (SELECT id from candidates WHERE last_name = 'Obama');
```

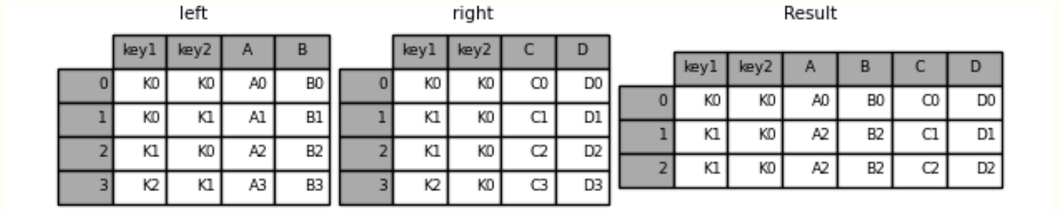
Two table grammar: JOINS

MUTATING JOINS: add new variables to one table from matching rows in another: inner join, left(outer) join, right(outer) join, and full(outer) join

INNER JOINS



Combine 2 tables on a common key value. 90% of the time this is an explicit inner join



left

	key1	key2	A	B
0	K0	K0	A0	B0
1	K0	K1	A1	B1
2	K1	K0	A2	B2
3	K2	K1	A3	B3

right

	key1	key2	C	D
0	K0	K0	C0	D0
1	K1	K0	C1	D1
2	K1	K0	C2	D2
3	K2	K0	C3	D3

Result

	key1	key2	A	B	C	D
0	K0	K0	A0	B0	C0	D0
1	K1	K0	A2	B2	C1	D1
2	K1	K0	A2	B2	C2	D2

PANDAS:

```
dfcwci.merge(dfcand, left_on="candidate_id", right_on="id")
```

SQL:

```
SELECT * FROM
    contributors JOIN candidates
ON contributors.candidate_id = candidates.id;
```

Warner Herzog

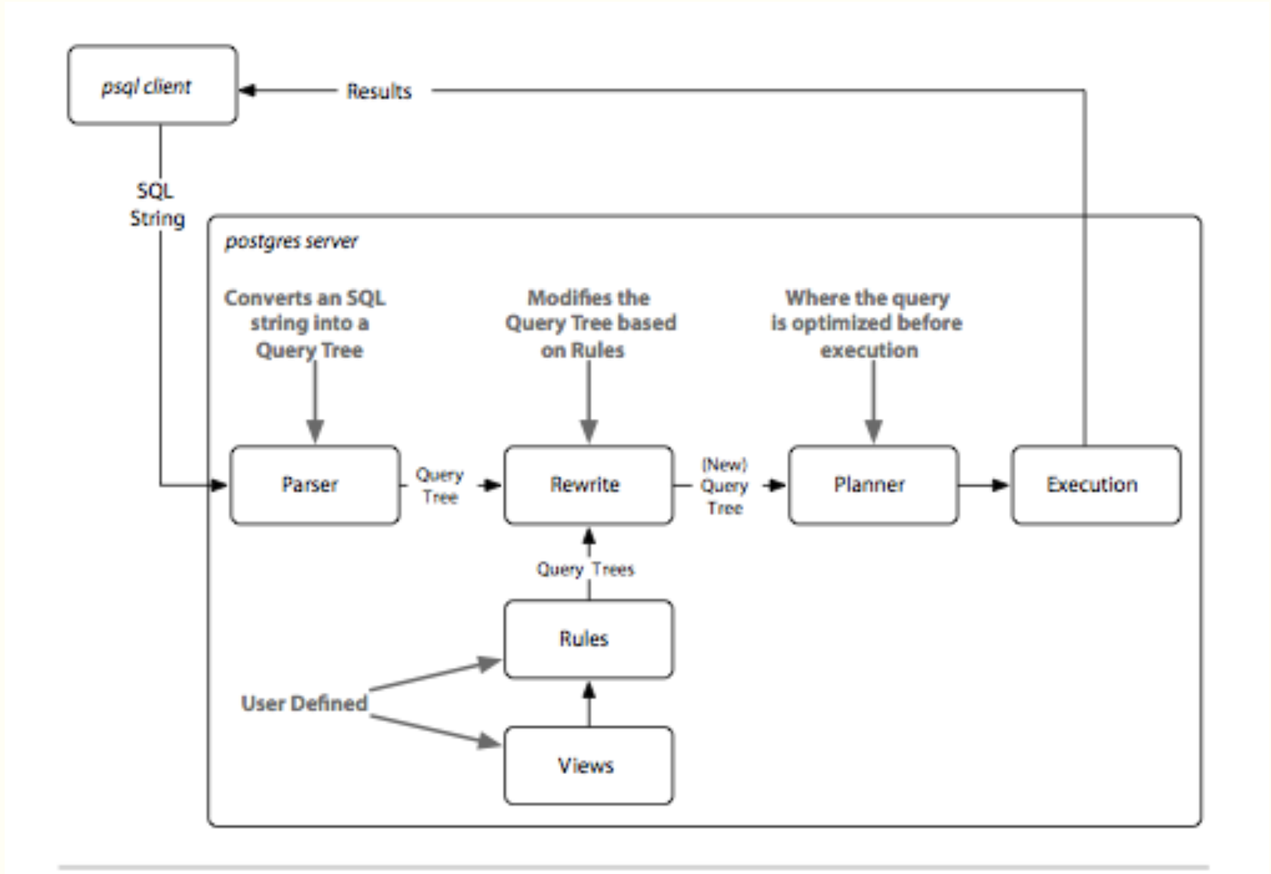


Diagram from 7 databases in 7 weeks, Pragmatic Programmers