# 61A Lecture 33

# Announcements

# Joining Tables

## Reminder: John the Patriotic Dog Breeder

```
create table parents as
select "abraham" as parent, "barack" as child union
select "abraham"          , "clinton"         union
select "delano"           , "herbert"         union
select "fillmore"         , "abraham"         union
select "fillmore"         , "delano"          union
select "fillmore"         , "grover"          union
select "eisenhower"       , "fillmore";
```

**Parents:**

| Parent | Child |
|---|---|
| abraham | barack |
| abraham | clinton |
| delano | herbert |
| fillmore | abraham |
| fillmore | delano |
| fillmore | grover |
| eisenhower | fillmore |

## Joining Two Tables

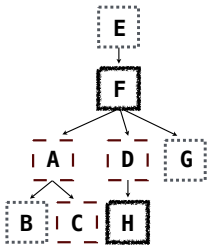
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
create table dogs as
  select "abraham" as name, "long" as fur union
  select "barack"         , "short"       union
  select "clinton"        , "long"        union
  select "delano"         , "long"        union
  select "eisenhower"     , "short"       union
  select "fillmore"       , "curly"       union
  select "grover"         , "short"       union
  select "herbert"        , "curly";

create table parents as
  select "abraham" as parent, "barack" as child union
  select "abraham"          , "clinton"         union
  ...;
```

Select the parents of curly-furred dogs

```
select parent from parents, dogs
              where child = name and fur = "curly";
```




(Demo)

# Aliases and Dot Expressions

## Joining a Table with Itself

Two tables may share a column name; dot expressions and aliases disambiguate column values

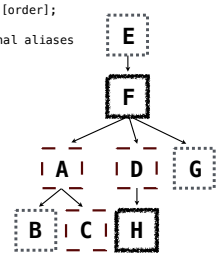
```
select [columns] from [table] where [condition] order by [order];
```

[table] is a comma-separated list of table names with optional aliases

Select all pairs of siblings

```
select a.child as first, b.child as second
  from parents as a, parents as b
  where a.parent = b.parent and a.child < b.child;
```

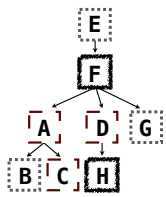
| First | Second |
|---|---|
| barack | clinton |
| abraham | delano |
| abraham | grover |
| delano | grover |




## Example: Grandparents

Which select statement evaluates to all grandparent, grandchild pairs?

1. `select a.grandparent, b.child from parents as a, parents as b where b.parent = a.child;`
2. `select a.parent, b.child from parents as a, parents as b where a.parent = b.child;`
3. `select a.parent, b.child from parents as a, parents as b where b.parent = a.child;`
4. `select a.grandparent, b.child from parents as a, parents as b where a.parent = b.child;`
5. None of the above



## Joining Multiple Tables

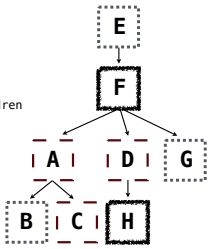
Multiple tables can be joined to yield all combinations of rows from each

```
create table grandparents as
  select a.parent as grandog, b.child as granpup
    from parents as a, parents as b
    where b.parent = a.child;
```

Select all grandparents with the same fur as their grandchildren

Which tables need to be joined together?

```
select grandog from grandparents, dogs as c, dogs as d
                where grandog = c.name and
                      granpup = d.name and
                      c.fur = d.fur;
```




## Example: Dog Triples

## Fall 2014 Quiz Question (Slightly Modified)

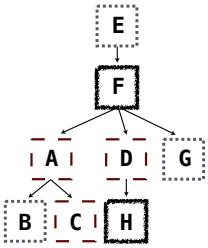
Write a SQL query that selects all possible combinations of three different dogs with the same fur and lists each triple in *inverse* alphabetical order

```
create table dogs as
  select "abraham" as name, "long" as fur union
  select "barack"         , "short"       union
  ...;

create table parents as
  select "abraham" as parent, "barack" as child union
  select "abraham"          , "clinton"          union
  ...;
```

Expected output:

```
delano|clinton|abraham
grover|eisenhower|barack
```




(Demo)

## Numerical Expressions

## Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
select [columns] from [table] where [expression] order by [expression];
                  [expression] as [name], [expression] as [name], ...
```

Combine values: +, −, *, /, %, and, or

Transform values: abs, round, not, −

Compare values: <, <=, >, >=, <>, !=, =

(Demo)

## String Expressions

## String Expressions

String values can be combined to form longer strings

```
sqlite> select "hello," || " world";
hello, world
```

Basic string manipulation is built into SQL, but differs from Python

```
sqlite> create table phrase as select "hello, world" as s;
sqlite> select substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) from phrase;
low
```

Strings can be used to represent structured values, but doing so is rarely a good idea

```
sqlite> create table lists as select "one" as car, "two,three,four" as cdr;
sqlite> select substr(cdr, 1, instr(cdr, ",")-1) as cadr from lists;
two
```

(Demo)