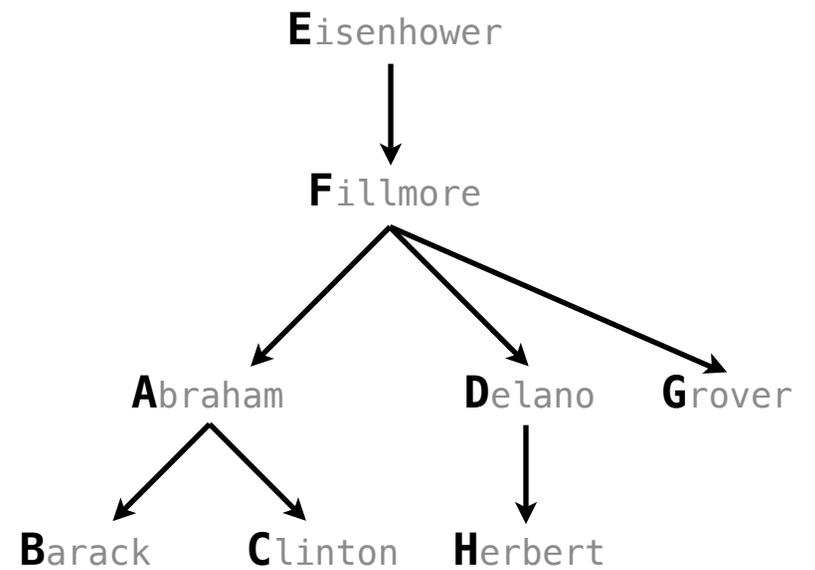


61A Lecture 31

Announcements

Joining Tables

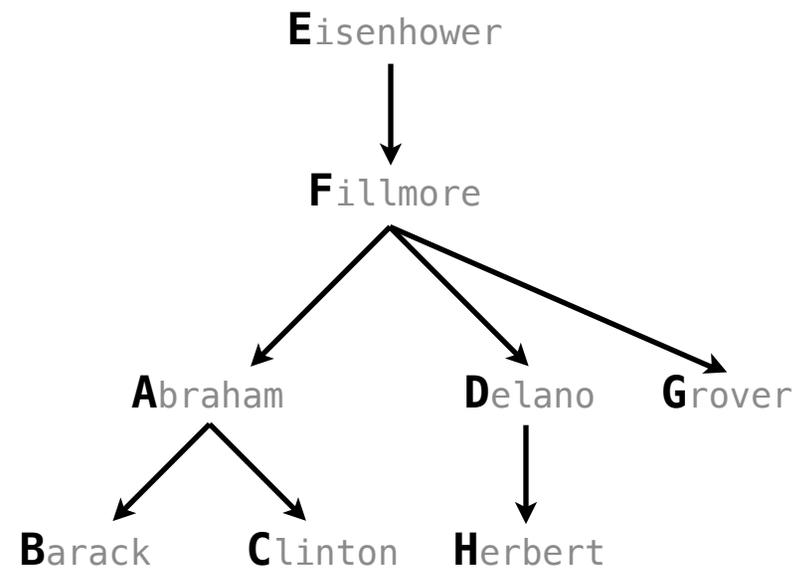
Reminder: John the Patriotic Dog Breeder



Reminder: John the Patriotic Dog Breeder



```
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";
```

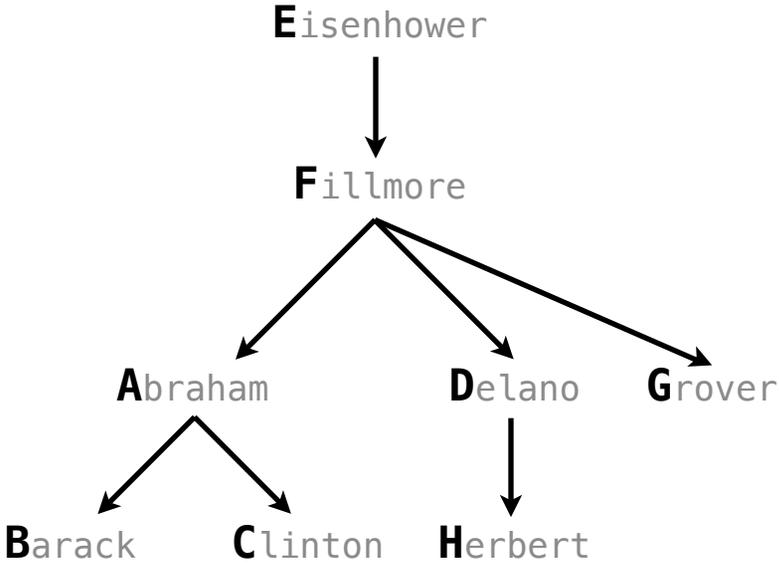


Reminder: John the Patriotic Dog Breeder



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";
```



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

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**

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
```

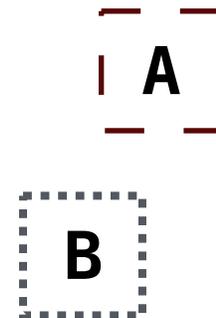


A

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**

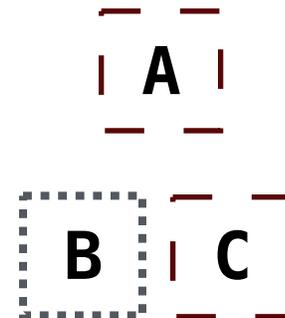
```
create table dogs as
  select "abraham" as name, "long" as fur union
  select "barack"      , "short"      union
```



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**

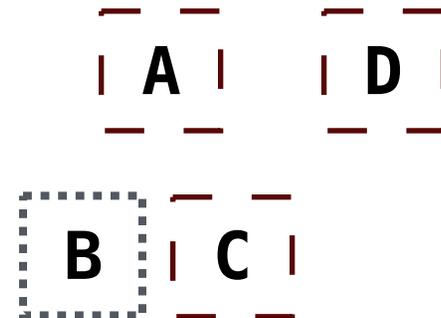
```
create table dogs as
  select "abraham" as name, "long" as fur union
  select "barack"      , "short"      union
  select "clinton"    , "long"       union
```



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
```



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
```



E



A **D**



B **C**

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
```

E

F

A | **D**

B | **C**

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
```

E

F

A

D

G

B

C

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";
```

E

F

A

D

G

B

C

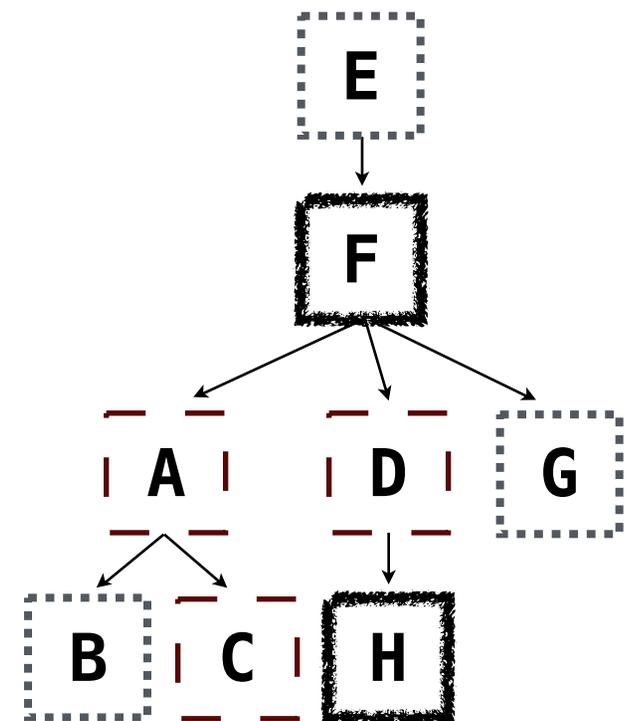
H

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
  ...;
```



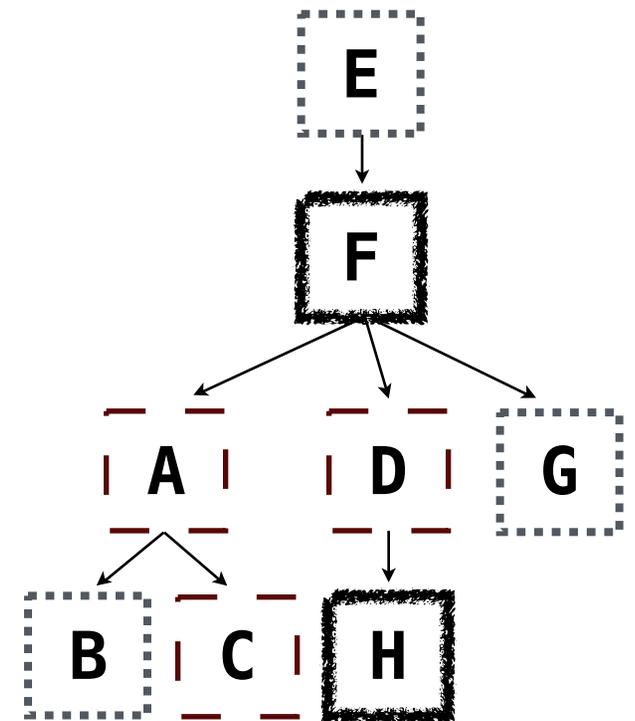
Joining Two Tables

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



Joining Two Tables

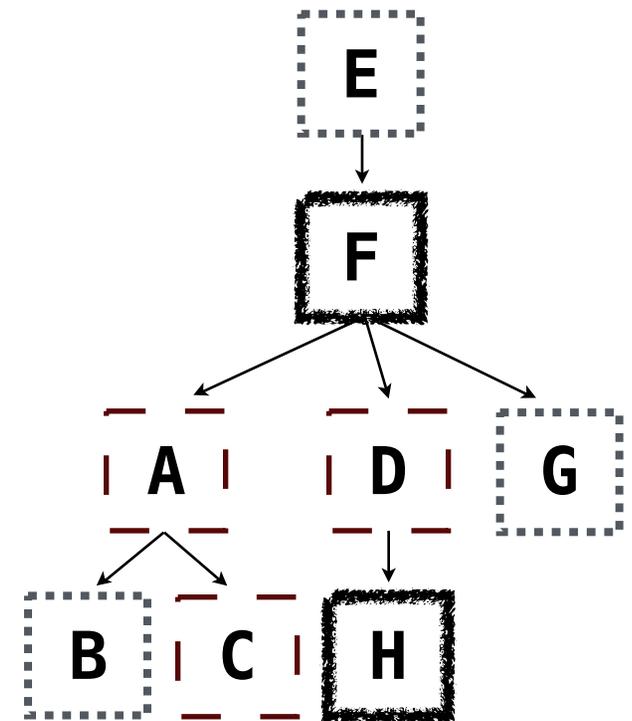
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
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";
```



Joining Two Tables

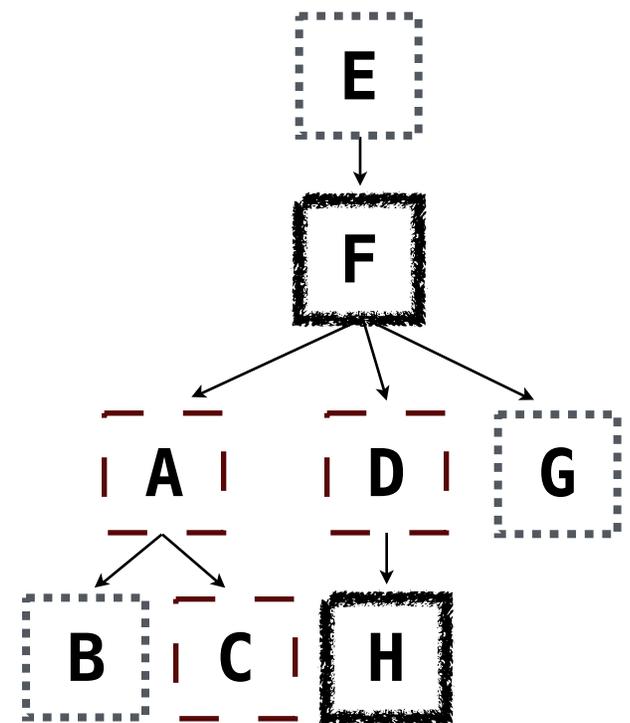
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
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";
```



Joining Two Tables

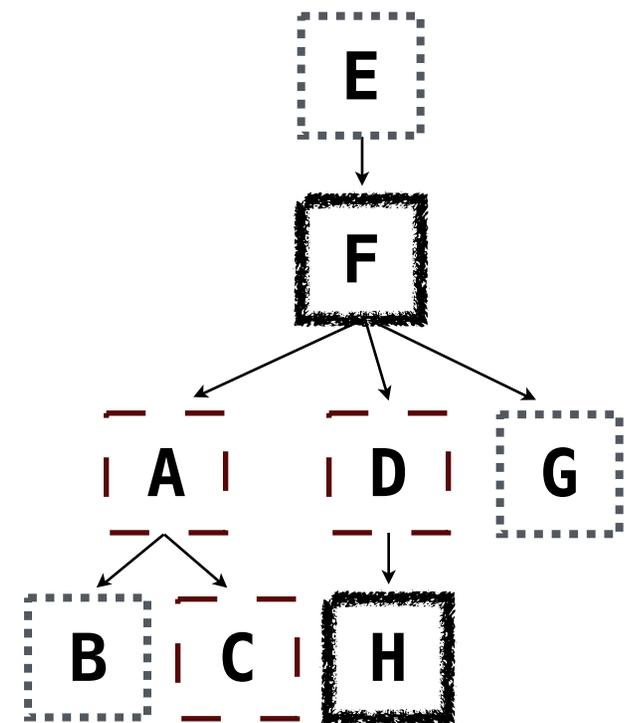
Two tables **A** & **B** are joined by a comma to yield all combos of a row from **A** & a row from **B**

```
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

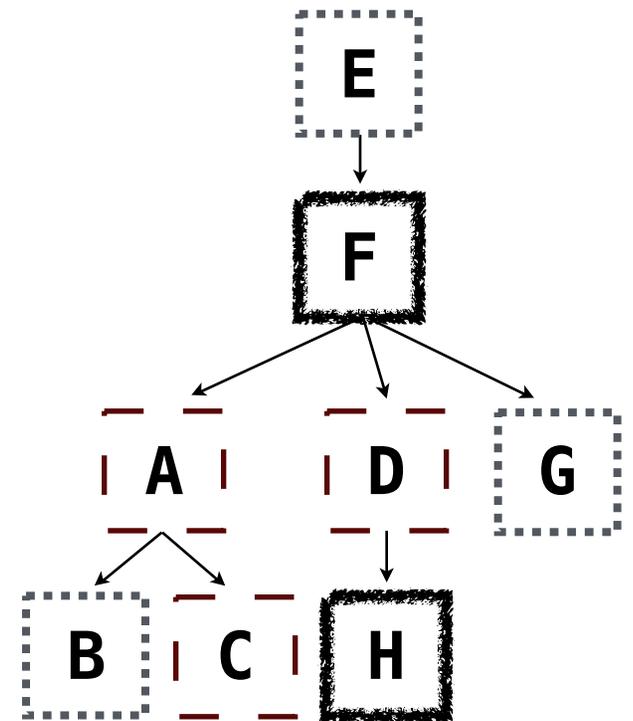
```
select parent from parents, dogs
  where child = name and fur = "curly";
```



(Demo)

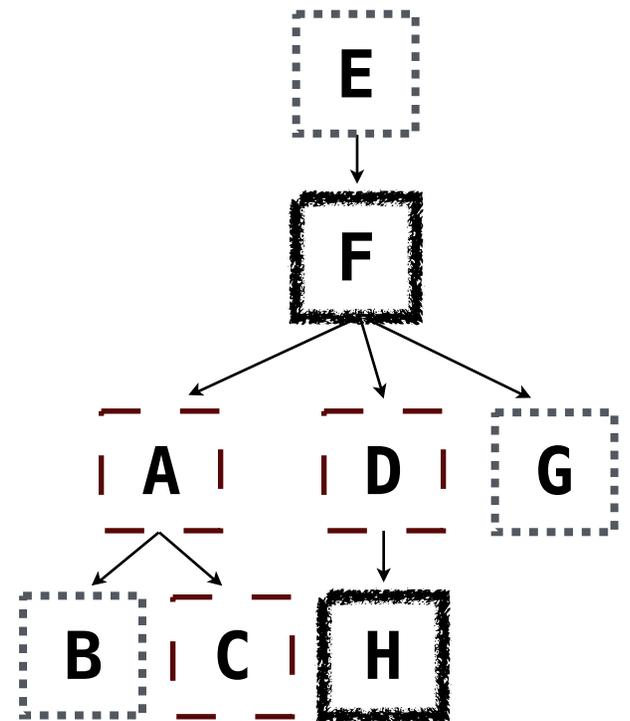
Aliases and Dot Expressions

Joining a Table with Itself



Joining a Table with Itself

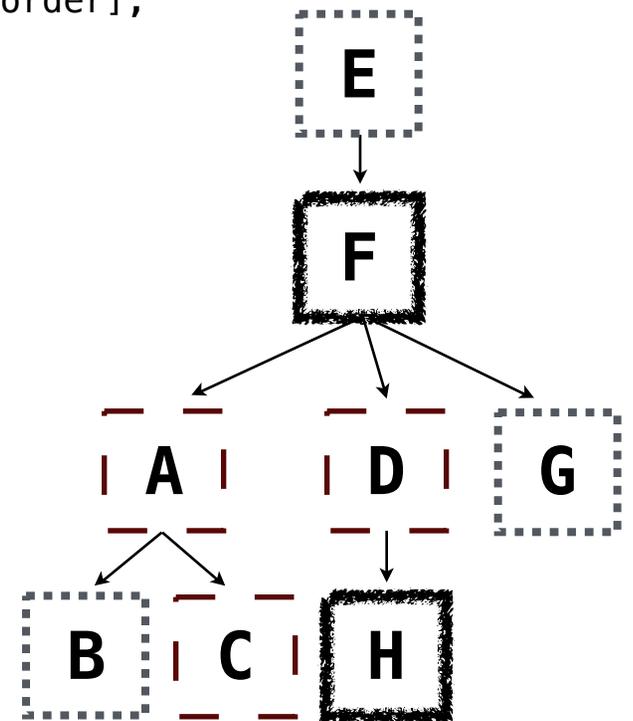
Two tables may share a column name; dot expressions and aliases disambiguate column values



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];
```

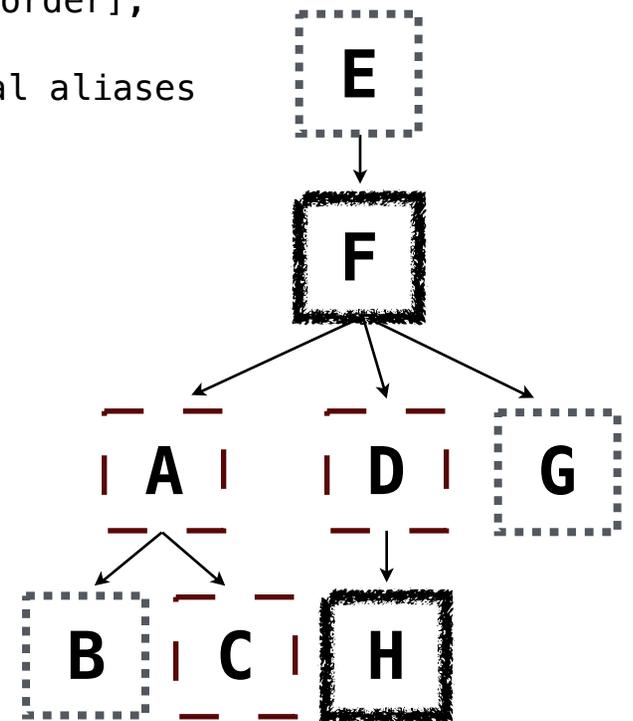


Joining a Table with Itself

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



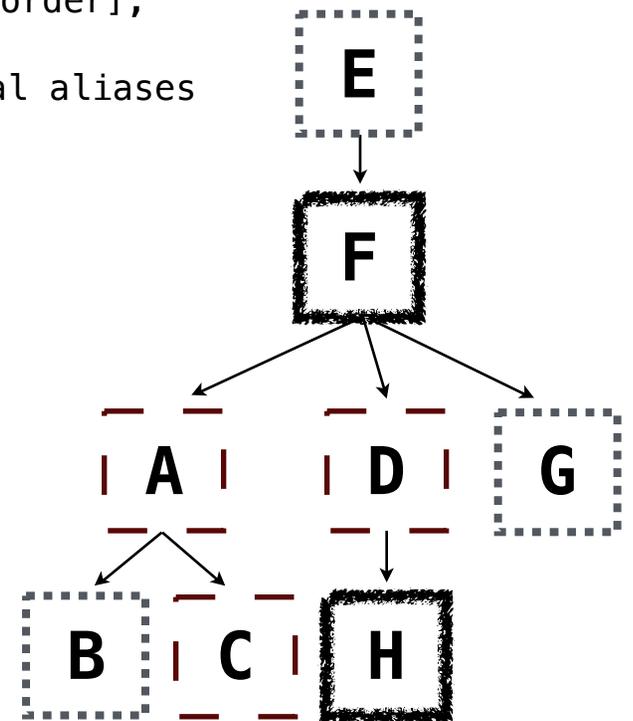
Joining a Table with Itself

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



Joining a Table with Itself

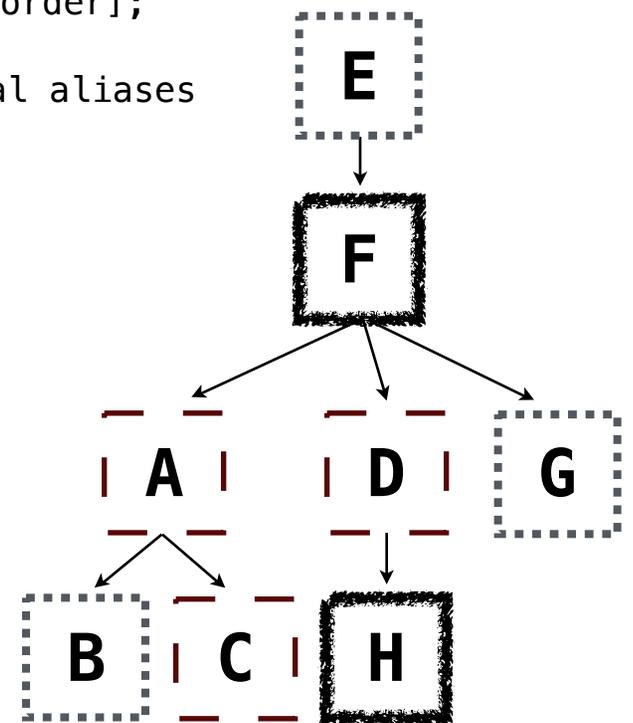
Two tables may share a column name; dot expressions and aliases disambiguate column values

```
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;
```



Joining a Table with Itself

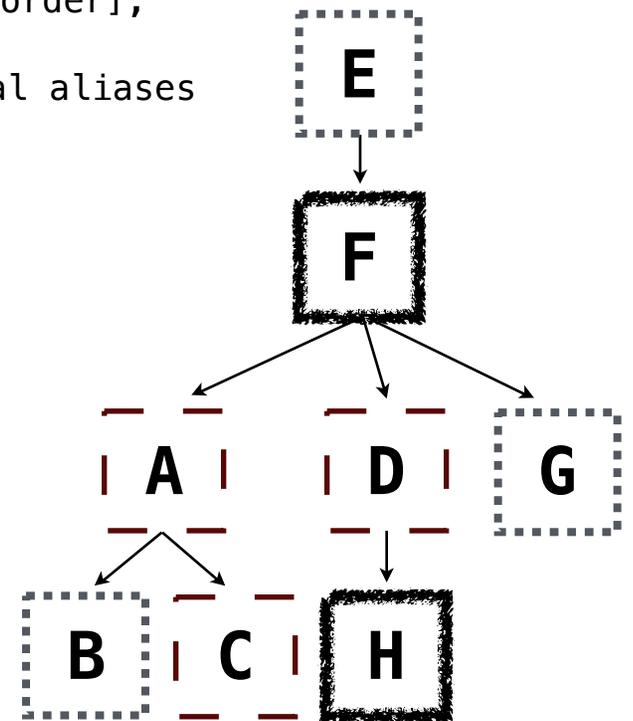
Two tables may share a column name; dot expressions and aliases disambiguate column values

```
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;
```



Joining a Table with Itself

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

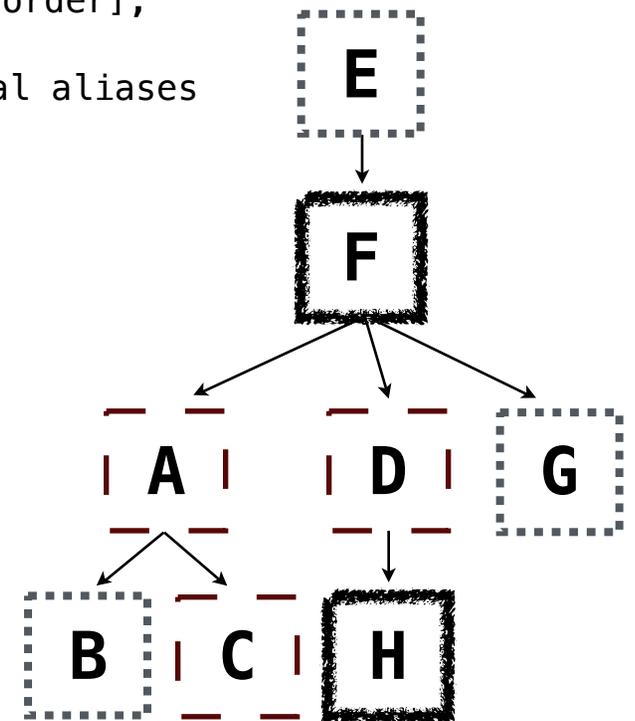
```
select [columns] from [table] where [condition] order by [order];
```

[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;
```

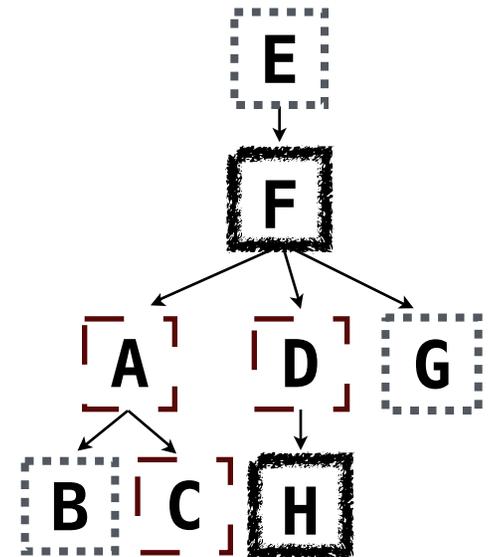
First	Second
barack	clinton
abraham	delano
abraham	grover
delano	grover



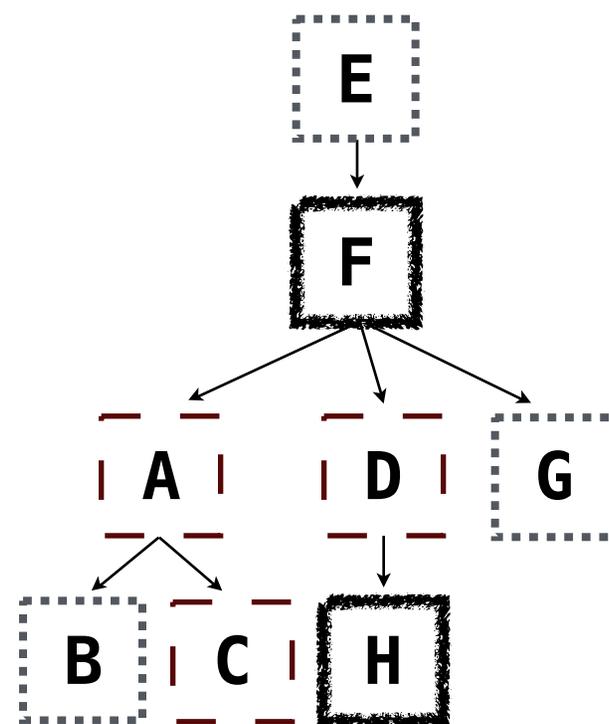
Example: Grandparents

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

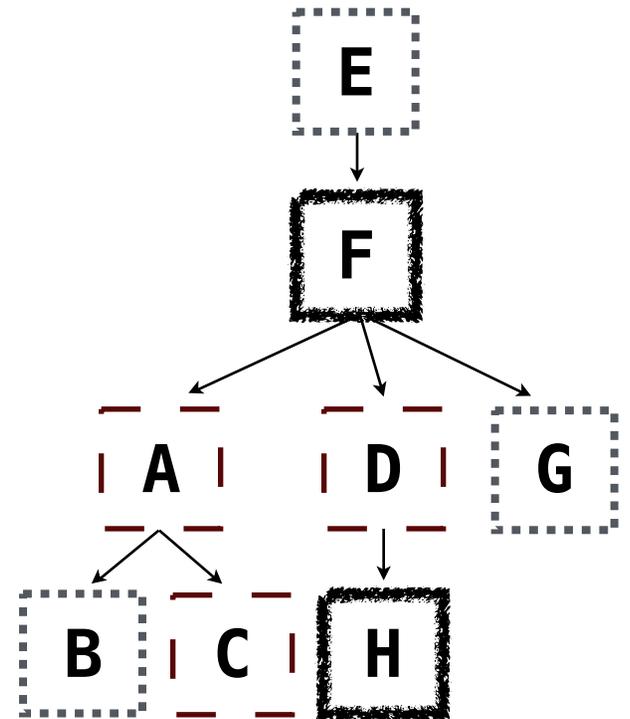


Joining Multiple Tables



Joining Multiple Tables

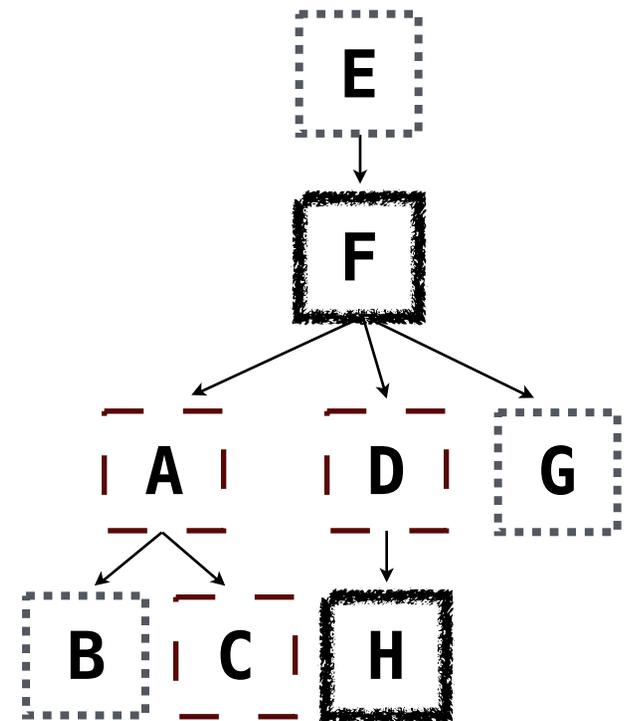
Multiple tables can be joined to yield all combinations of rows from each



Joining Multiple Tables

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

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

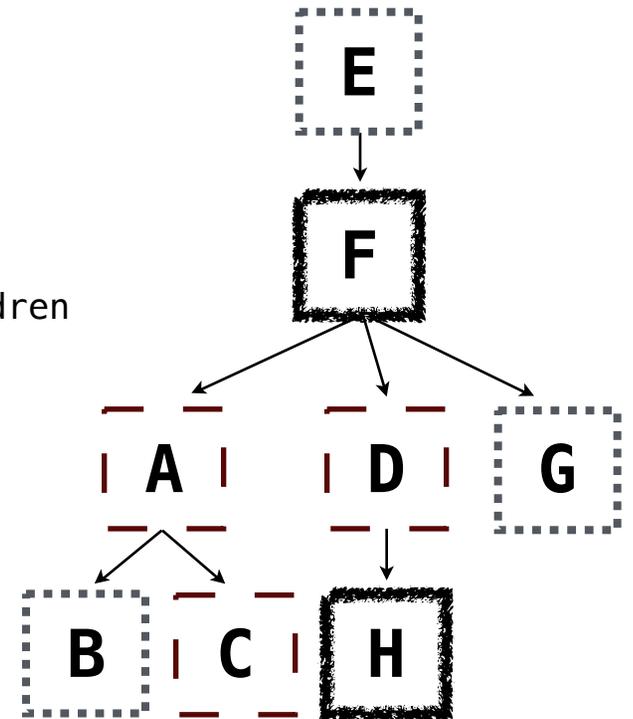


Joining Multiple Tables

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

```
create table grandparents as
select a.parent as granddog, 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



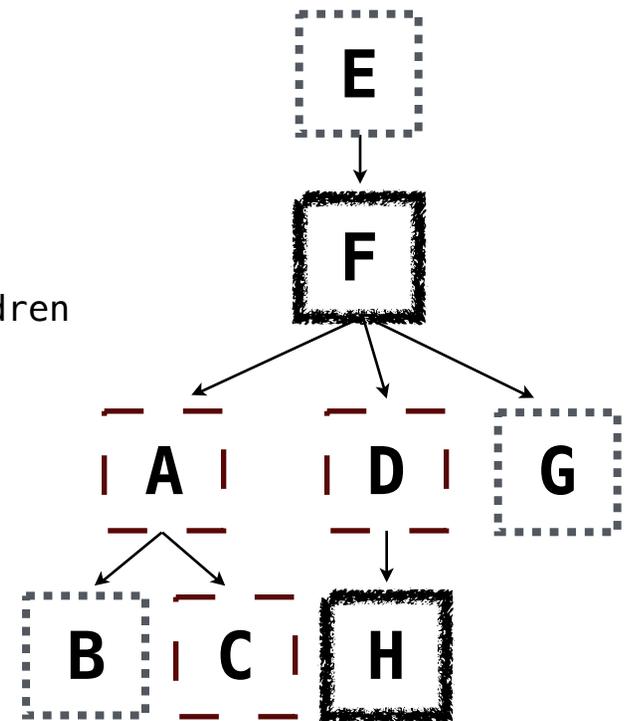
Joining Multiple Tables

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

```
create table grandparents as
select a.parent as granddog, 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?



Joining Multiple Tables

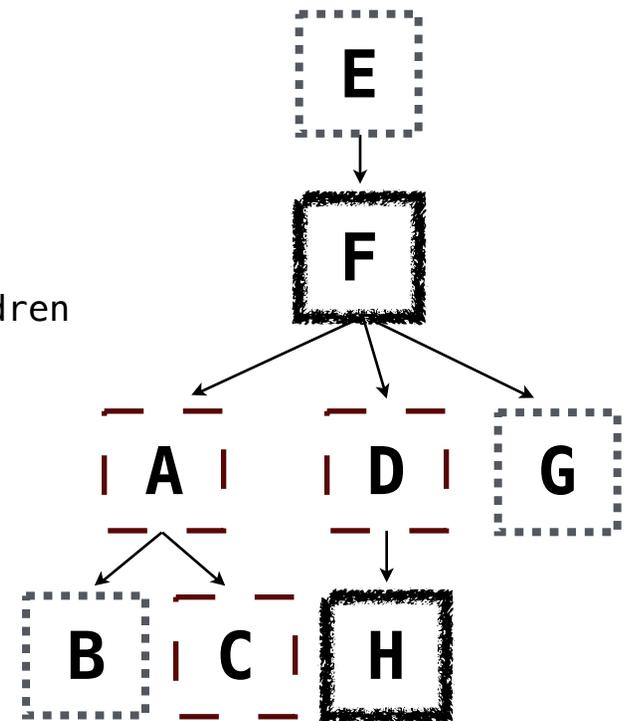
Multiple tables can be joined to yield all combinations of rows from each

```
create table grandparents as
select a.parent as granddog, 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 granddog from grandparents, dogs as c, dogs as d
where granddog = c.name and
granpup = d.name and
c.fur = d.fur;
```



Numerical Expressions

Numerical Expressions

Expressions can contain function calls and arithmetic operators

Numerical Expressions

Expressions can contain function calls and arithmetic operators

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

Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
[expression] as [name], [expression] as [name], ...
```

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

Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
[expression] as [name], [expression] as [name], ...
```

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

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

Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
[expression] as [name], [expression] as [name], ...
```

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

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

Transform values: abs, round, not, -

Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
[expression] as [name], [expression] as [name], ...
```

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

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

Transform values: abs, round, not, -

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

Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
[expression] as [name], [expression] as [name], ...
```

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

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

Transform values: abs, round, not, -

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

(Demo)

Example: Dog Triples

Fall 2014 Quiz Question

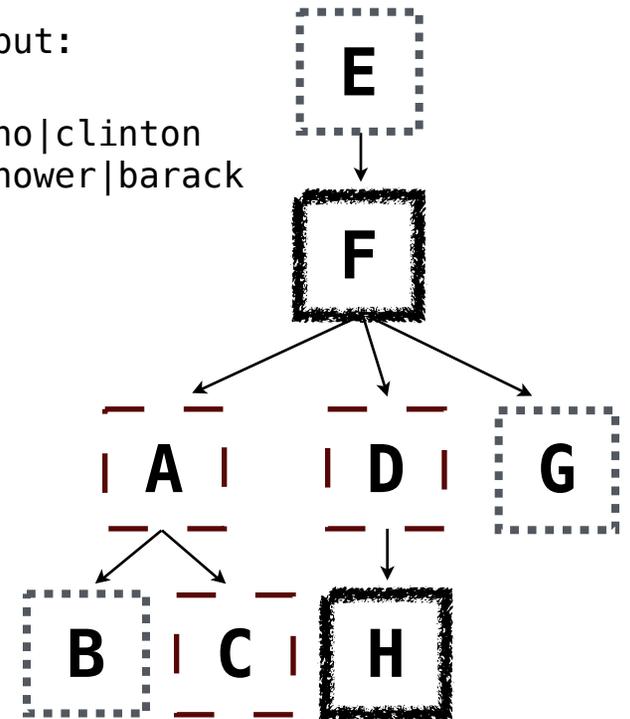
Write a SQL query that selects all possible combinations of three dogs with the same fur and lists them in order of increasing height

```
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"        union
select "delano"           , "jackson";
```

```
create table dogs as
select "abraham" as name, "long" as fur, 26 as height union
select "barack"   , "short"   , 52 union
select "clinton"  , "long"    , 47 union
select "delano"   , "long"    , 46 union
select "eisenhower" , "short" , 35 union
select "fillmore" , "curly"  , 32 union
select "grover"    , "short"  , 28 union
select "herbert"  , "curly"  , 31 union
select "jackson"  , "long"   , 43;
```

Expected output:

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



String Expressions

String Expressions

String values can be combined to form longer strings

String Expressions

String values can be combined to form longer strings



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

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

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;
```

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;
```

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
```

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
```

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

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;
```

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;
```

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
```

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
```

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

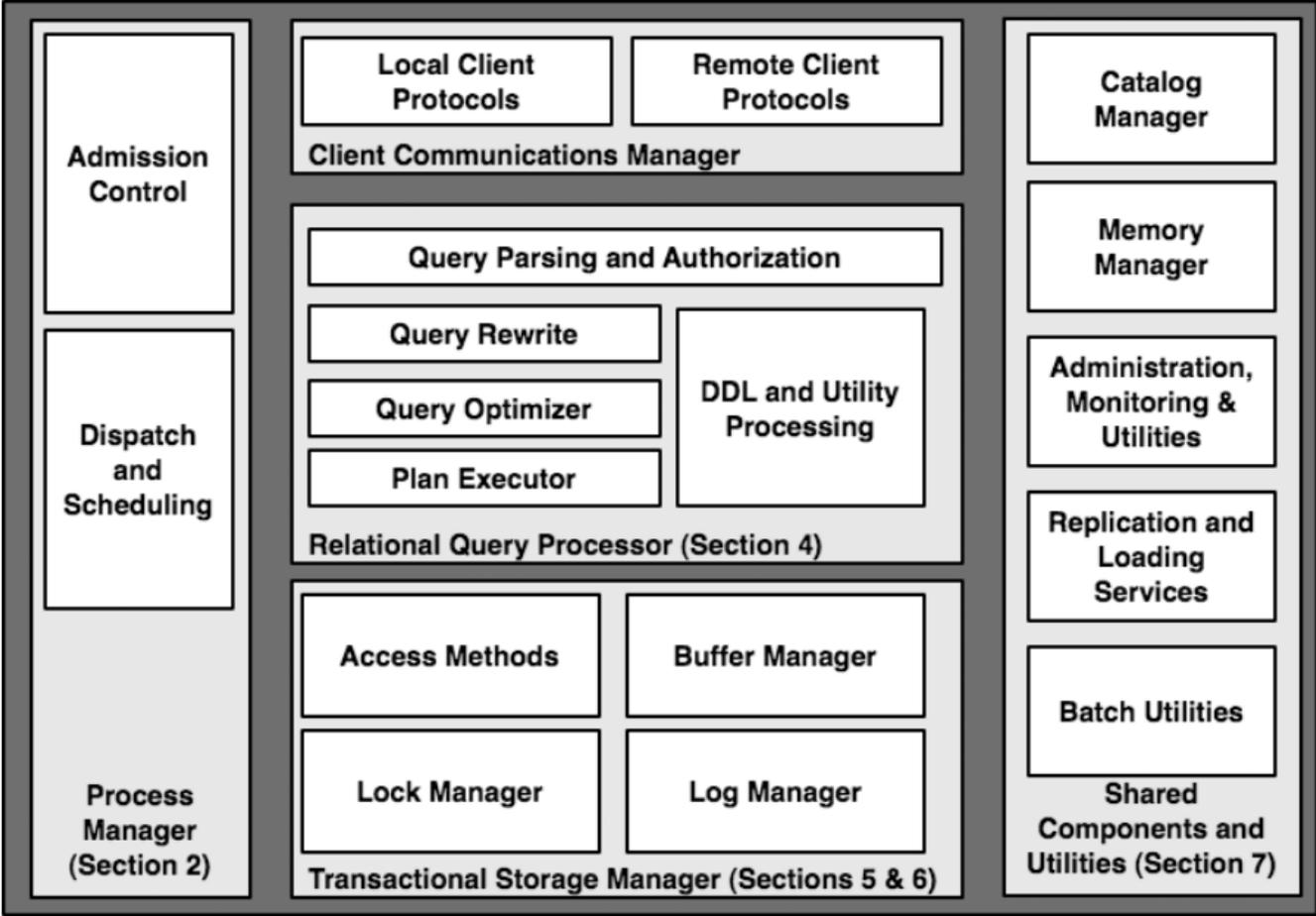


```
sqlite> create table lists as select "one" as car, "two,three,four" as cdr;  
sqlite> select substr(cdr, 1, instr(cdr, ",")-1) as cadr from lists;  
two
```

(Demo)

Database Management Systems

Database Management System Architecture



Architecture of a Database System by Hellerstein, Stonebreaker, and Hamilton

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Join all rows of parents to all rows of dogs, filter by `child = name` and `fur = "curly"`

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Join all rows of parents to all rows of dogs, filter by `child = name` and `fur = "curly"`

Join only rows of parents and dogs where `child = name`, filter by `fur = "curly"`

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Join all rows of parents to all rows of dogs, filter by `child = name` and `fur = "curly"`

Join only rows of parents and dogs where `child = name`, filter by `fur = "curly"`

Filter dogs by `fur = "curly"`, join result with all rows of parents, filter by `child = name`

Query Planning

The manner in which tables are filtered, sorted, and joined affects execution time

Select the parents of curly-furred dogs:

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

Join all rows of parents to all rows of dogs, filter by `child = name` and `fur = "curly"`

Join only rows of parents and dogs where `child = name`, filter by `fur = "curly"`

Filter dogs by `fur = "curly"`, join result with all rows of parents, filter by `child = name`

Filter dogs by `fur = "curly"`, join only rows of result and parents where `child = name`