Tables

Joining Tables

CREATE TABLE parents AS
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham" AS parent, "clinton" UNUNION
SELECT "delano" AS parent, "herbert" UNION
SELECT "fillmore" , "abraham" UNION
SELECT "fillmore", "delano" UNION
SELECT "fillmore", "grover" UNION
SELECT "eisenhower", "fillmore";

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";

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

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 "delano", "herbert" UNION
SELECT "fillmore", "abraham" UNION
SELECT "fillmore", "delano" UNION
SELECT "fillmore", "grover" UNION
SELECT "eisenhower", "fillmore";

Select all pairs of siblings

SELECT a.parent, b.child
WHERE a.parent = b.child AND a.child < b.child;

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 a.parent = b.child;

2. SELECT a.parent, b.child FROM parents AS a, parents AS b
WHERE b.parent = a.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.

```sql
CREATE TABLE grandparents AS
SELECT a.parent AS grandp, b.child AS grandpup
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?

```sql
SELECT grandp FROM grandparents, dogs AS c, dogs AS d
WHERE grandp = c.name AND
    grandpup = d.name AND
    c.fur = d.fur;
```

Which tables need to be joined together?

```sql
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;
```

Which tables need to be joined together?

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.

```sql
CREATE TABLE dogs AS
SELECT "abraham" AS name, "long" AS fur UNION
SELECT "barack" AS name, "short" AS fur UNION
...;
```

CREATE TABLE parents AS

```sql
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham", "clinton" UNION
...;
```

Expected output:

delano|clinton|abraham
grover|eisenhower|barack

Numerical Expressions

Expressions can contain function calls and arithmetic operators.

```sql
SELECT [columns] FROM [table] WHERE [expression] ORDER BY [expression];
```

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

Transform values: abs, round, not, -

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

(String Expressions)

String Expressions

String values can be combined to form longer strings.

```sql
sqlite> CREATE TABLE phrase AS SELECT "hello, world" AS s;
```

```sql
sqlite> SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1)
FROM phrase;
```

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

```sql
sqlite> CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;
```

```sql
sqlite> SELECT substr(cdr, 1, instr(cdr, ",")-1) AS cdr FROM lists;
```

(Demo)