Declarative Languages

Database Management Systems

Database management systems (DBMS) are important, heavily used, and interesting!

A table is a collection of records, which are rows that have a value for each column

The Structured Query Language (SQL) is perhaps the most widely used programming language

SQL is a declarative programming language

A table has columns and rows

A row has a value for each column

Latitude Longitude Name
38 122 Berkeley
42 71 Cambridge
45 93 Minneapolis

A column has a name and a type

Declarative Programming

In declarative languages such as SQL & Prolog:

- A "program" is a description of the desired result
- The interpreter figures out how to generate the result

In imperative languages such as Python & Scheme:

- A "program" is a description of computational processes
- The interpreter carries out execution/evaluation rules

In SQL:

- A select statement creates a new table, either from scratch or by projecting a table
- A create table statement gives a global name to a table
- Lots of other statements exist: analyze, delete, explain, insert, replace, update, etc.
- Most of the important action is in the select statement

SQL Overview

The SQL language is an ANSI and ISO standard, but DBMS's implement custom variants

Install sqlite (version 3.8.3 or later): http://sqlite.org/download.html
Use sqlite online: code.cs61a.org/sql

Today's theme:
Selecting Value Literals

A select statement always includes a comma-separated list of column descriptions. A column description is an expression, optionally followed by as and a column name:

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

Selecting literals creates a one-row table.

The union of two select statements is a table containing the rows of both of their results:

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

Naming Tables

SQL is often used as an interactive language. The result of a select statement is displayed to the user, but not stored. A create table statement gives the result a name:

```
create table [name] as [select statement];
```

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

Projecting Tables

A select statement can specify an input table using a from clause. A subset of the rows of the input table can be selected using a where clause. An ordering over the remaining rows can be declared using an order by clause. Column descriptions determine how each input row is projected to a result row:

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

```
select child from parents where parent = "abraham";
select parent from parents where parent > child;
```

Select Statements Project Existing Tables

In a select expression, column names evaluate to row values. Arithmetic expressions can combine row values and constants:

```
create table lift as
select 101 as chair, 2 as single, 2 as couple
union
select 102, 0, 3
union
select 103, 4, 1;
```

```
select chair, single + 2 * couple as total
from lift;
```

Arithmetic in Select Expressions

In a select expression, column names evaluate to row values. Arithmetic expressions can combine row values and constants:

```
create table lift as
select 101 as chair, 2 as single, 2 as couple
union
select 102, 0, 3
union
select 103, 4, 1;
```

```
select chair, single + 2 * couple as total
from lift;
```

<table>
<thead>
<tr>
<th>Chair</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>6</td>
</tr>
<tr>
<td>102</td>
<td>6</td>
</tr>
<tr>
<td>103</td>
<td>6</td>
</tr>
</tbody>
</table>
Discussion Question

Given the table that describes how to sum powers of 2 to form various integers:

<table>
<thead>
<tr>
<th>word</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>zero</td>
<td>0</td>
</tr>
<tr>
<td>one</td>
<td>1</td>
</tr>
<tr>
<td>two</td>
<td>2</td>
</tr>
<tr>
<td>four</td>
<td>4</td>
</tr>
<tr>
<td>eight</td>
<td>8</td>
</tr>
</tbody>
</table>

(A) Write a select statement for the word names of the powers of two.

(B) Write a select statement for the word and value for each integer.

CREATE TABLE ints AS

1. select "zero" as word, 0 as value
2. union
3. select "one", 1
4. union
5. select "two", 0, 2
6. union
7. select "three", 1, 2
8. union
9. select "four", 0, 0, 4
10. union
11. select "five", 1, 0, 4
12. union
13. select "six", 0, 2, 4
14. union
15. select "seven", 1, 2, 4
16. union
17. select "eight", 0, 0, 0, 8
18. union
19. select "nine", 1, 0, 0, 8

Joining Tables

Reminder: John the Patriotic Dog Breeder

CREATE TABLE parents AS

<table>
<thead>
<tr>
<th>name</th>
<th>parent</th>
<th>child</th>
</tr>
</thead>
<tbody>
<tr>
<td>abraham</td>
<td>barack</td>
<td></td>
</tr>
<tr>
<td>abraham</td>
<td>clinton</td>
<td></td>
</tr>
<tr>
<td>delano</td>
<td>herbert</td>
<td></td>
</tr>
<tr>
<td>fillmore</td>
<td>abraham</td>
<td></td>
</tr>
<tr>
<td>fillmore</td>
<td>delano</td>
<td></td>
</tr>
<tr>
<td>fillmore</td>
<td>grover</td>
<td></td>
</tr>
<tr>
<td>eisenhower</td>
<td>fillmore</td>
<td></td>
</tr>
</tbody>
</table>

Parents:

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 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

1. SELECT "abraham" AS name, "long" AS fur UNION
2. SELECT "barack", "short"
3. SELECT "clinton", "long"
4. SELECT "delano", "long"
5. SELECT "eisenhower", "short"
6. SELECT "fillmore", "curly"
7. SELECT "grover", "short"
8. SELECT "herbert", "curly"

CREATE TABLE parents AS

1. SELECT "abraham" AS parent, "barack" AS child UNION
2. SELECT "abraham", "clinton"
3. ...

Select the parents of curly-furred dogs:

SELECT parent FROM parents, dogs
WHERE child = name AND fur = 'curly';

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;

<table>
<thead>
<tr>
<th>First</th>
<th>Second</th>
</tr>
</thead>
<tbody>
<tr>
<td>barack</td>
<td>clinton</td>
</tr>
<tr>
<td>abraham</td>
<td>delano</td>
</tr>
<tr>
<td>abraham</td>
<td>grover</td>
</tr>
</tbody>
</table>

(Demo)
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 b.parent = a.child;
3. SELECT a.grandparent, b.child FROM parents AS a, parents AS b WHERE b.parent = a.child;
4. SELECT a.parent, b.child FROM parents AS a, parents AS b WHERE a.parent = b.child;
5. None of the above

Example: Dog Triples

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" AS name, "short" AS fur UNION
...;
CREATE TABLE parents AS
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham" AS parent, "clinton" AS child UNION
...;

Expected output:
delano|clinton|abraham
grover|eisenhower|barack

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" AS name, "short" AS fur UNION
...;
CREATE TABLE parents AS
SELECT "abraham" AS parent, "barack" AS child UNION
SELECT "abraham" AS parent, "clinton" AS child UNION
...;

Expected output:
delano|clinton|abraham
grover|eisenhower|barack

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: <, <=, >, >=, <>, !=, =
String Expressions

String values can be combined to form longer strings:

\[
\text{sqlite}\rangle \text{CREATE TABLE phrase AS SELECT "hello, world" AS s;}
\]

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

\[
\text{sqlite}\rangle \text{CREATE TABLE lists AS SELECT "one" AS car, "two,three,four" AS cdr;}
\]

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

\[
\text{sqlite}\rangle \text{SELECT substr(s, 4, 2) || substr(s, instr(s, " ")+1, 1) FROM phrase;}
\]

\[
\text{sqlite}\rangle \text{SELECT substr(cdr, 1, instr(cdr, ",")-1) AS cadr FROM lists;}
\]