**Declarative Languages**

A select statement always includes a comma-separated list of column descriptions.

A select statement may include a comma-separated list of table names.

A column description is an expression, optionally followed by an alias and a column name.

**Structured Query Language (SQL)**

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

### Selecting Value Literals

A select statement always includes a comma-separated list of column descriptions.

A column description is an expression, optionally followed by an alias and a column name.

**SQL Overview**

The SQL language 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.

**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.

**Getting Started with SQL**

Install sqlite (version 3.8.3 or later): http://sqlite.org/download.html

Use sqlite online: http://sqliteonline.com
Select Statements Project Existing 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 resulting 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], ...
- select (subselect) from (table) where (condition) order by (order);
- select child from parents where parent = "curly"
- select parent from parents where parent = "child"

Discussion Question

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

<table>
<thead>
<tr>
<th>power</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
</tr>
</tbody>
</table>

(a) Write a select statement for a two-column table of the word and value for each integer:

(b) Write a select statement for the word and value for all integers:

<table>
<thead>
<tr>
<th>word</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>six</td>
<td>6</td>
</tr>
<tr>
<td>five</td>
<td>5</td>
</tr>
<tr>
<td>three</td>
<td>3</td>
</tr>
<tr>
<td>two</td>
<td>2</td>
</tr>
<tr>
<td>one</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Aliases and Dot Expressions

CREATE table a

CREATE table b

CREATE table c

CREATE table d

CREATE table e

CREATE table f

SELECT a.name FROM a
SELECT b.name AS "short" FROM b
SELECT c.name AS "long" FROM c
SELECT d.name AS "fillmore" FROM d
SELECT e.name WHERE e.name = "fillmore"

SELECT a.name, b.name AS "short", c.name AS "long", d.name AS "fillmore" FROM a, b, c, d

SELECT a.name FROM a
SELECT b.name AS "short" FROM b
SELECT c.name AS "long" FROM c
SELECT d.name AS "fillmore" FROM d
SELECT e.name WHERE e.name = "fillmore"

SELECT a.name, b.name AS "short", c.name AS "long", d.name AS "fillmore" FROM a, b, c, d

Joining Tables

Joining Two Tables

Two tables A & B are joined by a common to yield all couples of a row from A & a row from B.

CREATE table A

CREATE table B

CREATE table C

CREATE table D

CREATE table E

CREATE table F

SELECT a.name FROM A
SELECT b.name AS "short" FROM B
SELECT c.name AS "long" FROM C
SELECT d.name AS "fillmore" FROM D
SELECT e.name WHERE e.name = "fillmore"

SELECT a.name, b.name AS "short", c.name AS "long", d.name AS "fillmore" FROM a, b, c, d

Joining a Table with Itself

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

SELECT (aliases) FROM (table) WHERE (condition) ORDER BY (order);

A table is a comma-separated list of table names with optional aliases, select all pairs of columns:

SELECT a.column AS first, b.column AS second
WHERE a.column < b.column
ORDER BY a.column, b.column;

Arithmetic in Select Expressions

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

CREATE table T

SELECT a.column AS first, b.column AS second
WHERE a.column < b.column
ORDER BY a.column, b.column;
Example: Grandparents

Which select statement evaluates to all grandparent, grandchild pairs?

1. select a.grandparent, b.child from parents a, parents b where a.parent = b.child;
2. select a.parent, b.child from parents a, parents b where a.parent = b.child;
3. select a.parent, b.child from parents a, parents b where b.parent = a.child;
4. select a.grandparent, b.child from parents a, parents 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

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" UNION
SELECT "grover" UNION
SELECT "eisenhower" UNION
SELECT "clinton" 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
```

Numerical Expressions

Expressions can contain function calls and arithmetic operators

```
expression AS [name], expression AS [name], ...
```

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

Transform values: abs, round, not, -

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

String Expressions

String values can be combined to form larger strings

```
sqlite> CREATE table phrase AS SELECT "hello, world"
```

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

```
sqlite> CREATE table lists AS SELECT "one, two, three, four"
```

Expressions

```
expression AS [name], expression AS [name], ...
```

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

Transform values: abs, round, not, -

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