CS61A: SQL Basic

Lin Sheng

January 2022

1 Database management systems

Database management systems are built up by tables. A table is a collection of records, which are rows that have a value for each column. A column has a name and a type. A row has a value for each column.

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

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.

```
create table Cities as
    select 38 as latitude, 122 as longitude, "Berkeley" as name union
    select 42, 71, "Cambridge";

select "west coast" as region, name from cities where longitude >= 115 union
select "other" as region, name from cities where longitude < 115;
```

2 SQL overview

- 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.

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.

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];
```

A **select** statement can specify an input table using a **from** clause.

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

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.

```
select * from parents; # '*' represents all columns
```

Important: When you union together a bunch of select statements, you get no guarantee about the order of the result.

3 Arithmetic

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

4 Joining tables

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

5 Aliases and dot expressions

Two tables may share a column name; dot expressions and aliases disambiguate column values.
[table] is a comma-separated list of table names with optional aliases.

```
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;
```

6 Numerical expressions

Expressions can contain function calls and arithmetic operators.

- Combine values: +, -, *, /, %, and, or
- Transform values: abs, round, not, -
- Compare values: <, <=, >, >=, =, ! =, <>

7 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.

8 Aggregation

An aggregate function in the [columns] clause computes a value from a group of rows.

```
create table animals as
    select "dog" as kind, 4 as legs, 20 as weight union
    select "cat" as kind, 4 as legs, 22 as weight union
    select "parrot" as kind, 2 as legs, 6 as weight;
> select max(legs) from animals;
4
> select count(legs) from animals:
3
> select count(distinct legs) from animals:
2
> select avg(legs) from animals:
16
```

An aggregate function also selects a row in the table, which may be meaningful.

```
    select max(weight), kind from animals;
    # It will get a row with column "kind" and "max(weight)"
      which are in the same row before.
    select avg(weight), kind from animals;
    # The "kind" here is not meaningful.
```

9 Groups

Rows in a table can be grouped, and aggregation is performed on each group.

```
select [column] from [table] group by [expression] having [expression];
```

The number of groups is the number of unique values of an expression.

```
select legs, max(weight) from animals group by legs;
```

A **having** clause filters the set of groups that are aggregated. A **where** clause filters individual rows.

```
select weight/legs, count(*) from animals group by weight/legs having count(*)>1;
```

10 Table

10.1 Create and drop table

Examples:

```
    create table numbers (n, note);
    create table numbers (n UNIQUE, note);
    create table numbers (n, note DEFAULT "No comment");
    drop table if exists numbers;
```

10.2 Modifying table

For a table t with two columns, to insert into one column: **INSERT INTO t(column) VALUES (value)**; to insert into both columns: **INSERT INTO t VALUES (value0, value1)**.

```
sqlite> create table primes(n, prime);
sqlite> drop table if exists primes;
sqlite> select * from primes;
Error: no such table: primes
sqlite> create table primes(n UNIQUE, prime DEFAULT 1);
sqlite> INSERT INTO primes VALUES (2, 1), (3, 1);
sqlite> select * from primes;
2|1
3|1
sqlite> INSERT INTO primes(n) VALUES (4) (5) (6);
sqlite> INSERT INTO primes(n) SELECT n+6 FROM primes;
```

10.3 Update

Update sets all entries in certain columns to new values, just for some subset of rows.

```
sqlite> UPDATE primes SET prime=0 WHERE n>2 AND n%2=0;
```

10.4 Delete

Delete removes some or all rows from a table.

```
sqlite> DELETE FROM primes WHERE prime=0;
sqlite> DELETE FROM primes; # delete all rows in primes
# The result is an empty table.
```