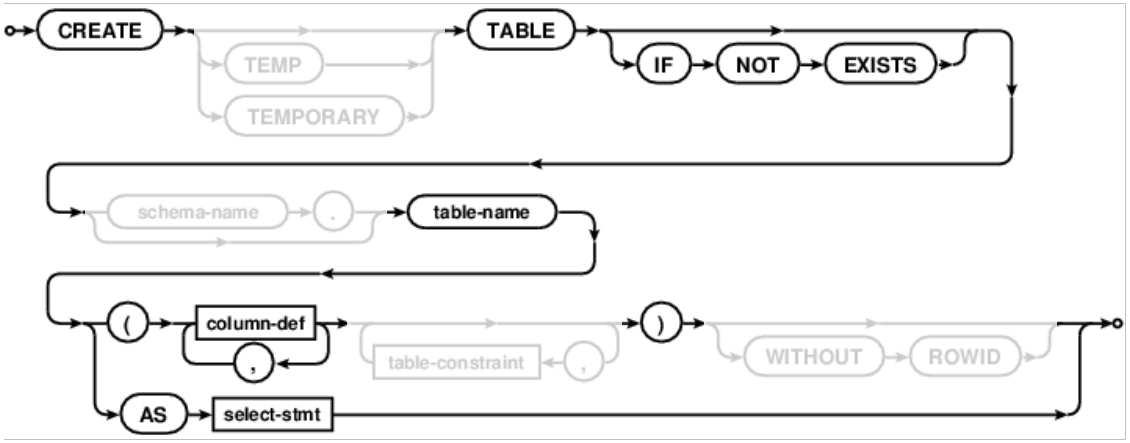
Databases

Announcements

Create Table and Drop Table

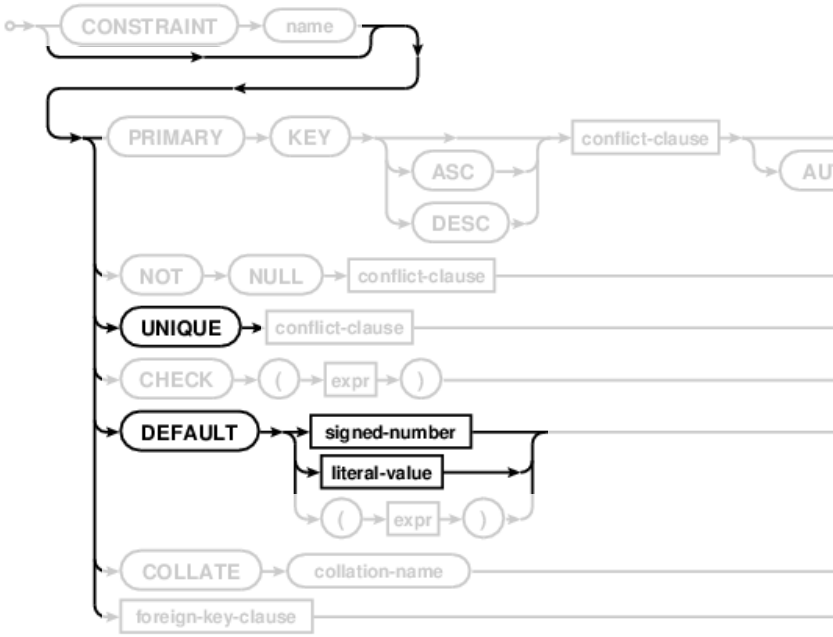
Create Table

CREATE TABLE expression syntax:



column-def:

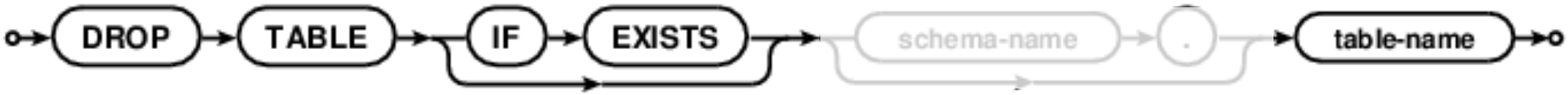
column-constraint:



Examples:

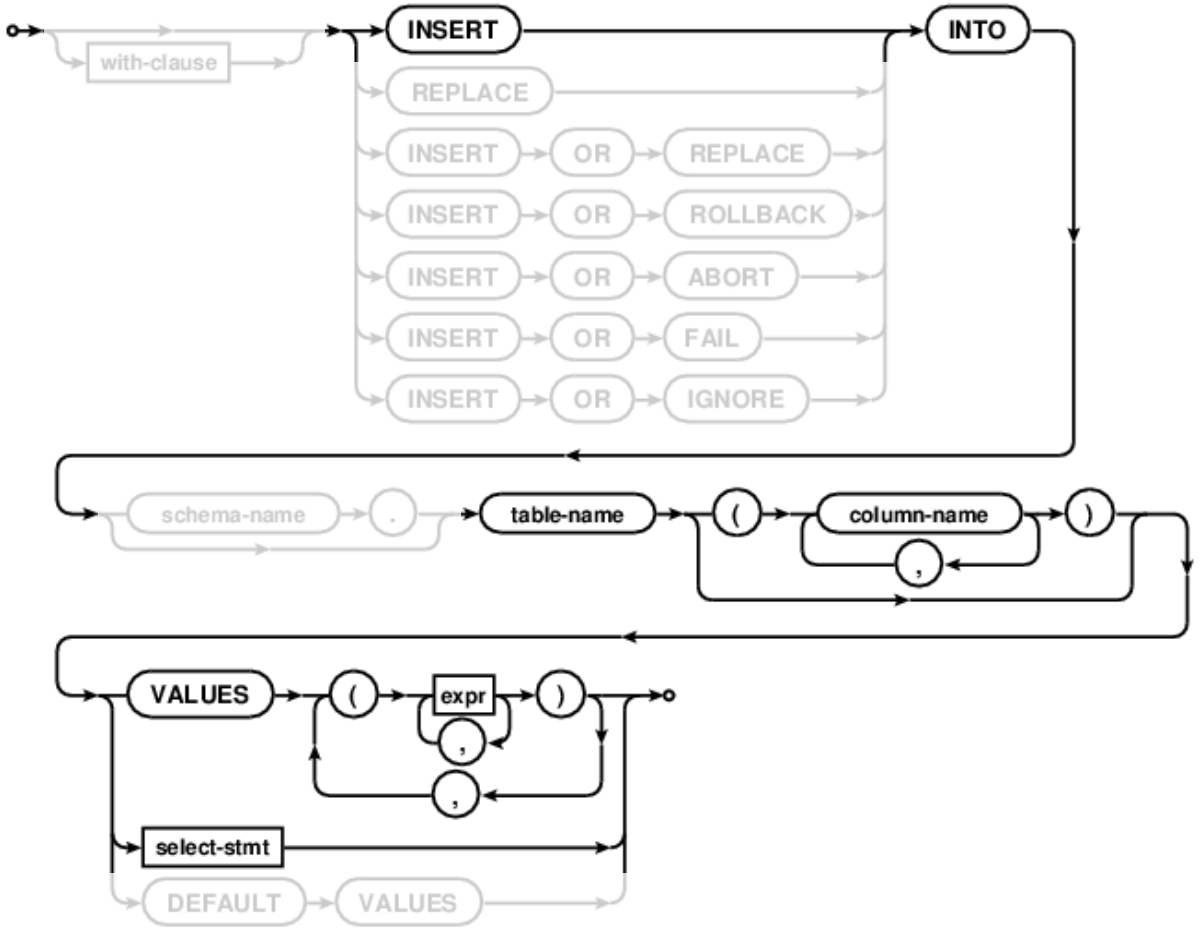
```
CREATE TABLE numbers (n, note);
CREATE TABLE numbers (n UNIQUE, note);
CREATE TABLE numbers (n, note DEFAULT "No comment");
```

Drop Table



Modifying Tables

Insert



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);
```

(Demo)

```
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> select * from primes;
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), (7);
sqlite> select * from primes;
2|1
3|1
4|1
5|1
6|1
7|1
```

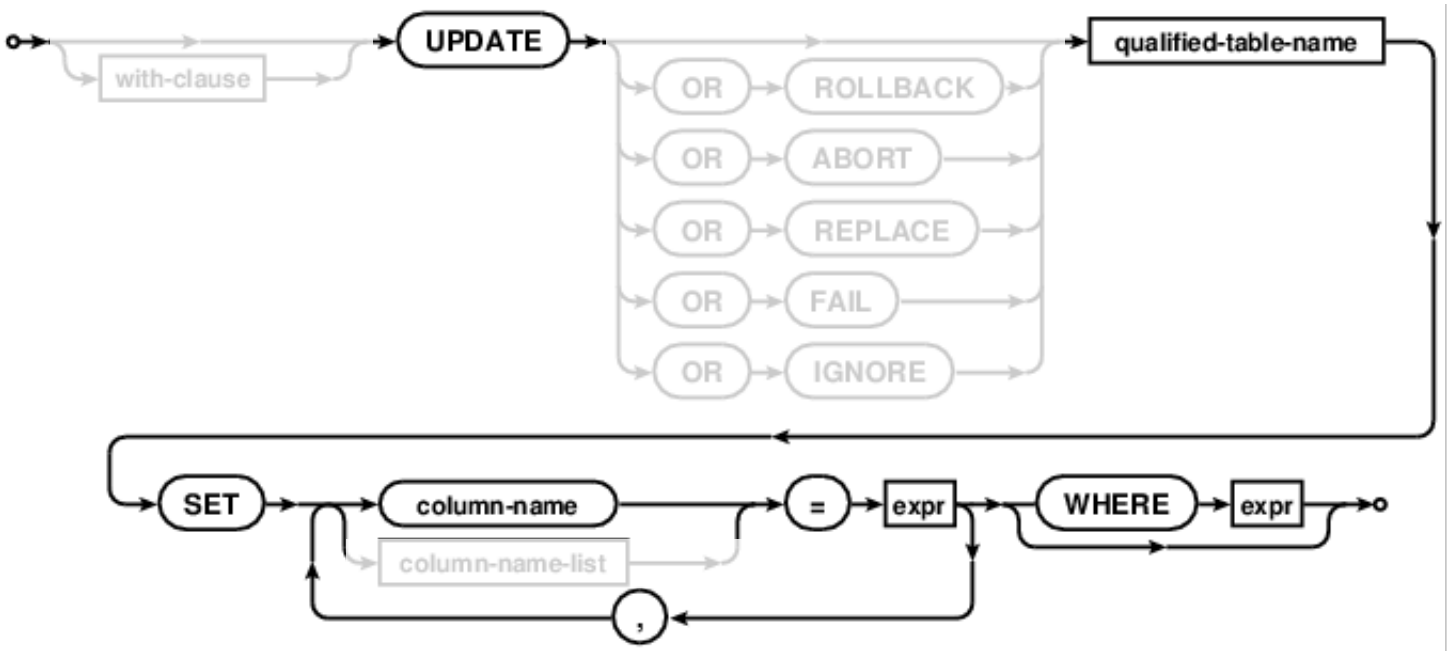
have to fix it later

```
sqlite> INSERT INTO primes(n) SELECT n+6 FROM primes;
sqlite> select * from primes;
2|1
3|1
4|1
5|1
6|1
7|1
8|1
9|1
10|1
11|1
12|1
13|1
```

```
sqlite> INSERT INTO primes(n) SELECT n+12 FROM primes;
sqlite> select * from primes;
2|1
3|1
4|1
5|1
6|1
7|1
8|1
9|1
10|1
11|1
12|1
13|1
14|1
15|1
...
```

```
sqlite> UPDATE primes SET prime=0 WHERE n>2 AND n%2=0;
sqlite> UPDATE primes SET prime=0 WHERE n>3 AND n%3=0;
sqlite> UPDATE primes SET prime=0 WHERE n>5 AND n%5=0;
```

Update

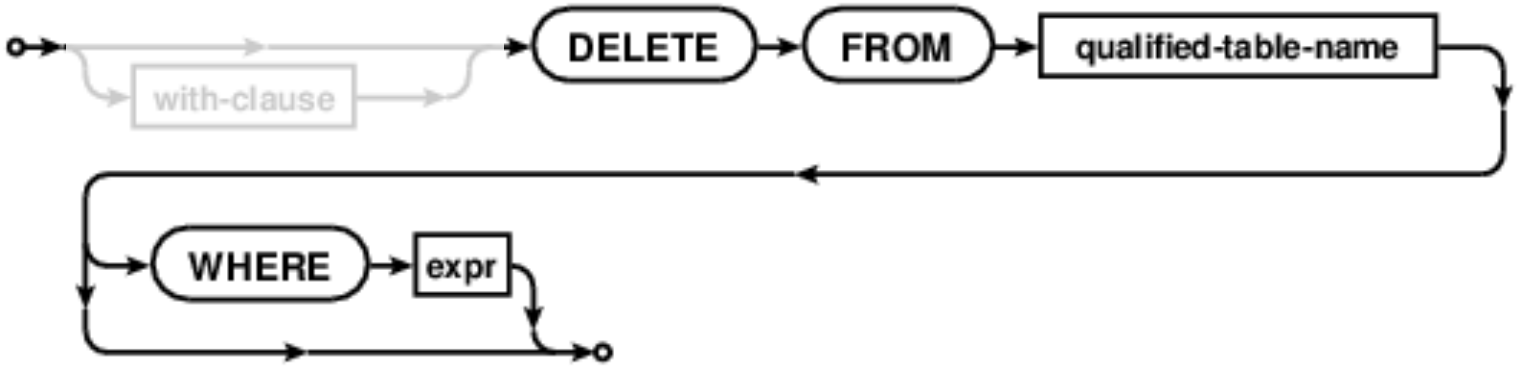


Update sets all entries in certain columns to new values, just for some subset of rows.

(Demo)

```
sqlite> DELETE FROM primes WHERE prime=0;
```

Delete



Delete removes some or all rows from a table.

(Demo)

Delete all rows -> The table will exist (empty table), different from drop table

```
import sqlite3

db = sqlite3.Connection("n.db")
db.execute("CREATE TABLE nums AS SELECT 2 UNION SELECT 3;")
db.execute("INSERT INTO nums VALUES (?), (?), (?);", range(4, 7))
print(db.execute("SELECT * FROM nums;").fetchall())
db.commit()
```

create / read a database

numbers in range(4, 7) take the place of ?

.execute method returns an object called cursor, which has a method .fetchall() which will return a tuple

.commit commits changes to the original database

Python and SQL

```
~/lec$ python3 ex.py
[(2,), (3,), (4,), (5,), (6,)]
~/lec$ ls n.db
n.db
~/lec$ sqlite3 n.db
SQLite version 3.19.3 2017-06-27 16:48:08
Enter ".help" for usage hints.
sqlite> SELECT * FROM nums;
2
3
4
5
6
```

(Demo)

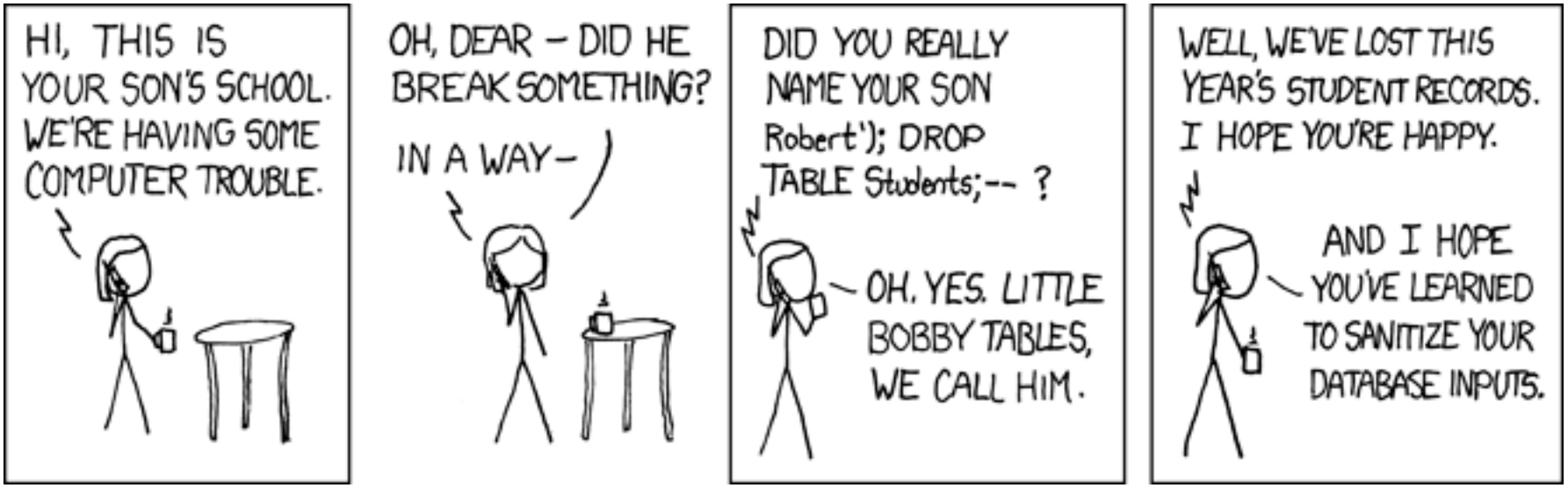
SQL Injection Attack

```
name = "Robert'); DROP TABLE Students; --"
cmd = "INSERT INTO Students VALUES ('" + name + "');"
db.executescript(cmd)
```

don't use +

.execute runs one statement at a time; .executescript runs all

A Program Vulnerable to a SQL Injection Attack




```
name = "Robert'); DROP TABLE Students; --"
~~cmd = "INSERT INTO Students VALUES ('" + name + "');"~~
~~db.executescript(cmd)~~   db.execute("INSERT INTO Students VALUES (?)", [name])
```

```
INSERT INTO Students VALUES ('Robert'); DROP TABLE Students; --');
~~INSERT INTO Students VALUES ('Robert'); DROP TABLE Students; --');~~
```

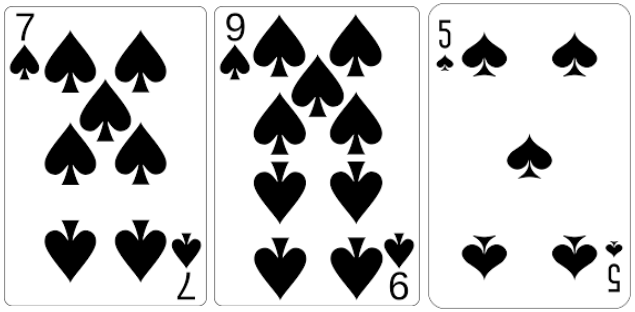
https://xkcd.com/327/

Database Connections

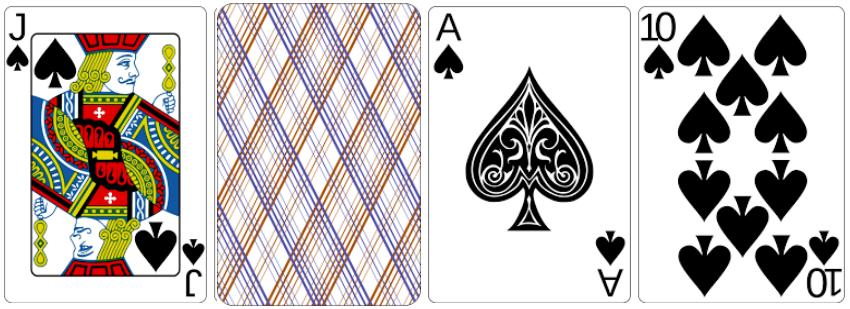
Multiple programs can be connected to a database at the same time, and thus inserting and reading values into / from the same tables.

Casino Blackjack

Player:



Dealer:




```
Dealing...
Hit? n
Player 16 and Dealer 18

Dealing...
Hit? y
Hit? y
Player went bust!

Dealing...
Hit? y
Hit? n
Dealer went bust!

Dealing...
Hit? n
Player 21 and Dealer 18

Dealing...
Hit?
```

```
lec — sqlite3 cards.db — 85×22
sqlite> select * from cards where who <> "Discard";
4|Player
9|Dealer
Q|Player
sqlite> UPDATE cards SET card="A" WHERE who="Player" and card=4;
sqlite> select * from cards where who <> "Discard";
A|Player
9|Dealer
Q|Player
sqlite>
```

2 programs accessing the database at the same time

(Demo)