sql public

sql builder

sql string builder for node - supports PostgreSQL, mysql, Microsoft SQL Server, Oracle and sqlite dialects.

Building SQL statements by hand is no fun, especially in a language which has clumsy support for multi-line strings.

So let's build it with JavaScript.

Maybe it's still not fun, but at least it's *less not fun*.

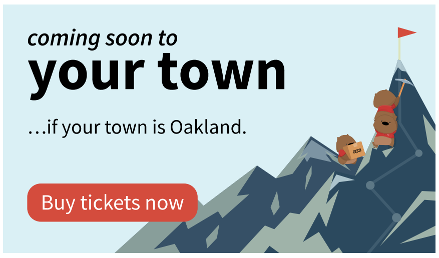
build passing

# install

```
$ npm install sql
```

# use

```
//require the module
var sql = require('sql');

//(optionally) set the SQL dialect
sql.setDialect('postgres');
//possible dialects: mssql, mysql, postgres (default), sqlite

//first we define our tables
var user = sql.define({
  name: 'user',
  columns: ['id', 'name', 'email', 'lastLogin']
});

var post = sql.define({
  name: 'post',
  columns: ['id', 'userId', 'date', 'title', 'body']
});

//now let's make a simple query
var query = user.select(user.star()).from(user).toQuery();
console.log(query.text); //SELECT "user".* FROM "user"

//something more interesting
var query = user
    .select(user.id)
    .from(user)
    .where(
      user.name.equals('boom').and(user.id.equals(1))
    ).or(
      user.name.equals('bang').and(user.id.equals(2))
    ).toQuery();

//query is parameterized by default
console.log(query.text); //SELECT "user"."id" FROM "user" WHERE ((("user"."name
```



```
npm install sql
```

how? learn more

brianc published a month ago

0.70.0 is the latest of 92 releases

github.com/brianc/node-sql

MIT

Collaborators

Stats

991 downloads in the last day

5 443 downloads in the last week

26 342 downloads in the last month

65 open issues on GitHub

7 open pull requests on GitHub

Try it out

Test sql in your browser.

Keywords

None

Dependencies (2)

lodash, sliced

Dependents

sql-generate, anydb-sql, @matteo.collina/timeseri
pg, sql-query-builder, dbcrawler, shop-keeper, trip
cg_model, odata-to-sql, pg-jobs, sqlmagic, salty-du
forerunner-postgres-store, seql, backbeam-server,
bass-sql, clay-sequelize, lowladb-node-postgresql,

```
console.log(query.values); //['boom', 1, 'bang', 2]

//queries can be named
var query = user.select(user.star()).from(user).toNamedQuery('user.all');
console.log(query.name); //'user.all'

//how about a join?
var query = user.select(user.name, post.body)
  .from(user.join(post).on(user.id.equals(post.userId))).toQuery();

console.log(query.text); //'SELECT "user"."name", "post"."body" FROM "user" INN

//this also makes parts of your queries composable, which is handy

var friendship = sql.define({
  name: 'friendship',
  columns: ['userId', 'friendId']
});

var friends = user.as('friends');
var userToFriends = user
  .leftJoin(friendship).on(user.id.equals(friendship.userId))
  .leftJoin(friends).on(friendship.friendId.equals(friends.id));

//and now...compose...
var friendsWhoHaveLoggedInQuery = user.from(userToFriends).where(friends.lastLo
//SELECT * FROM "user"
//LEFT JOIN "friendship" ON ("user"."id" = "friendship"."userId")
//LEFT JOIN "user" AS "friends" ON ("friendship"."friendId" = "friends"."id")
//WHERE "friends"."lastLogin" IS NOT NULL

var friendsWhoUseGmailQuery = user.from(userToFriends).where(friends.email.like
//SELECT * FROM "user"
//LEFT JOIN "friendship" ON ("user"."id" = "friendship"."userId")
//LEFT JOIN "user" AS "friends" ON ("friendship"."friendId" = "friends"."id")
//WHERE "friends"."email" LIKE %1

//Using different property names for columns
//helpful if your column name is long or not camelCase
var user = sql.define({
  name: 'user',
  columns: [{
      name: 'id'
    }, {
      name: 'state_or_province',
      property: 'state'
    }
  ]
});

//now, instead of user.state_or_province, you can just use user.state
console.log(user.select().where(user.state.equals('WA')).toQuery().text);
// "SELECT "user".* FROM "user" WHERE ("user"."state_or_province" = $1)"
```

There are a **lot** more examples included in the **test/dialects** folder. We encourage you to read through them if you have any questions on usage!

findhit-sequelize, pine2e, sqoosel, caramel, relatior
node-pg-jobs, sqlbox, fixr, voltron-postgres-adapte
fixr-compiled, worm, ectypes-downstairs, pg-dal

Hired is hiring. View more…

# from the command line

You can use the **sql-generate module** to automatically generate definition files from a database instance. For example, running `node-sql-generate --dsn "mysql://user:password@host/database"` will generate something similar to:

```
// autogenerated by node-sql-generate v0.0.1 on Tue May 21 2013 01:04:12 GMT-07
var sql = require('sql');

/**
 * SQL definition for database.bar
 */
exports.bar = sql.define({
    name: 'bar',
    columns: [
        'id',
        'foo_id'
    ]
});

/**
 * SQL definition for database.foo
 */
exports.foo = sql.define({
    name: 'foo',
    columns: [
        'id',
        'field_1',
        'foo_bar_baz'
    ]
});

/**
 * Adding a column to an existing table:
 */
var model = sql.define({ name: 'foo', columns: [] });
model.addColumn('id');

// If you try to add another column "id", node-sql will throw an error.
// You can suppress that error via:
model.addColumn('id', { noisy: false });
```

Read the module's documentation for more details.

# contributing

We **love** contributions.

node-sql wouldn't be anything without all the contributors and collaborators who've worked on it. If you'd like to become a collaborator here's how it's done:

1. fork the repo
2. `git pull https://github.com/(your_username)/node-sql`
3. `cd node-sql`
4. `npm install`
5. `npm test`

At this point the tests should pass for you. If they don't pass please open an issue with the output or you can even send me an email directly. My email address is on my github profile and also on every commit I contributed in the repo.

Once the tests are passing, modify as you see fit. *Please* make sure you write tests to cover your modifications. Once you're ready, commit your changes and submit a pull request.

As long as your pull request doesn't have completely off-the-wall changes and it does have tests we will almost always merge it and push it to npm

If you think your changes are too off-the-wall, open an issue or a pull-request without code so we can discuss them before you begin.

Usually after a few high-quality pull requests and friendly interactions we will gladly share collaboration rights with you.

After all, open source belongs to everyone.

# license

MIT