

PostgreSQL

Common Data Types:

INTEGER, FLOAT

VARCHAR(n): varying length character string of maximum length n

DATE: A date attribute in a DBMS-specific format.

Sample Use Case:

Create a table of students holding student IDs and names. Create another table that shows which student IDs are taking which courses:

```
CREATE TABLE Student(id INTEGER, name VARCHAR(20));
```

```
CREATE TABLE Takes(id INTEGER, course VARCHAR(20));
```

Data Manipulation

Add a new student:

```
INSERT INTO Student (id, name) VALUES (003841201, 'Zingaro', 'Daniel');
```

Update rows in a table (set a certain student's name based on id):

```
UPDATE Student SET name = "Gilfoyle" WHERE id = '1337';
```

Delete students named "Bob Jones" from the table:

```
DELETE FROM Student WHERE name = 'Bob Jones';
```

Find everyone taking CSC309:

PostgreSQL (cont)

```
SELECT name FROM Student, Takes WHERE Student.id = Takes.id AND Takes.course = 'CSC309';
```

PHP commands

```
$conn = $db_connect("$connect_string");
$result = $pg_query($conn, $query);
while ($row = $pg_fetch_array($result)) { do something }
```

If we're going through a list of students, `$row[id]` is equivalent to `Student.id` in above example.

HTTP Commands

```
<!-- GET a specified resource,
DELETE a specified resource, HEAD
get the response headers (without
associated data), PUT accompanying
data at the specified location,
POST accompanying data as a
subordinate to the specified
location. !-->
```

AUTHENTICATION REQUEST:

```
GET /~arnold/309/phttp/hello.html
HTTP/1.1
```

Host: www.cs.toronto.edu

Authorization: Basic

YXJub2xkOnByb2Y=

YXJub2xkOnByb2Y is a Base64 encoding of `userid:password`.

CSS Example

```
// selected navigation bar, do
stuff to it
nav {
    position: absolute; top:
52px; left:0;
width: 160px; height: 700px;
}
// now each individual button
nav a {
display: block;
border-color: blue;
border-style: solid;
background-color: #ACE;
text-align: center;
white-space: nowrap;
padding: 10px;
margin-bottom: 0.25em;
color:black;
}
```

Vote App (vote.js - Express)

```
require('./static-
content/lib/constants.js'); //
defines wwPort and wwWsPort
var express = require('express');
var app = express();
var votes = { yes:0 , no:0 } ;
// Web sockets to broadcast results
var WebSocketServer =
require('ws').Server
```



By **bogoSort**

cheatography.com/bogosort/

Not published yet.

Last updated 14th April, 2018.

Page 1 of 3.

Sponsored by **ApolloPad.com**

Everyone has a novel in them. Finish Yours!

<https://apollopad.com>

Vote App (vote.js - Express) (cont)

```
,wss = new
WebSocketServer({port: wwWsPort});
wss.on('close', function() {
    console.log('disconnected');
});
wss.broadcast = function(){
    for(let ws of
this.clients){
        ws.send(JSON.string
ify(votes));
    }
}
wss.on('connection', function(ws) {
    ws.send(JSON.stringify(vote
s));
});
var bodyParser = require('body-
parser');
app.use(bodyParser.json()); //
support json encoded bodies
app.use(bodyParser.urlencoded({
extended: true })); // support
encoded bodies
app.use(express.static('static-
content'));
app.put('/api/vote/:value/',
function (req, res) {
    var voteValue = req.params.value;
    if(voteValue=="yes")votes.yes++;
    if(voteValue=="no")votes.no++;

    console.log("PUT:"+voteValue);
    console.log("Total Votes so
far:"+votes);
```

Vote App (vote.js - Express) (cont)

```
var result = { result: "ok" ,
currentVotes: votes};
console.log(JSON.stringify(result))
;
res.json(result);
wss.broadcast();
});
app.listen(wwPort, function () {
    console.log('Example app
listening on port '+wwPort);
});
```

Vote App (vote.html - React, Sockets, REST)

```
<script type="text/javascript">
<script type="text/javascript">
    var socket;
    $(function(
    ){
        //
        socket = new
        WebSocket("ws://cslinux.utm.utoront
o.ca:10001");
        alert(global.wwWsURL);
        soc
        ket = new
        WebSocket(global.wwWsURL);
        so
        cket.onopen = function (event) {
            -
            console.log("connected");
        }
        ;
        so
        cket.onclose = function (event) {
            -
            alert("closed code:" +
            event.code + " reason:"
            +event.reason + "
            wasClean:"+event.wasClean);
        }
        ;
```

Vote App (vote.html - React, Sockets, REST) (cont)

```
so
cket.onmessage = function (event) {
    -
    var
    votes=JSON.parse(event.data);
    $("#VoteCount").html("Yes:"+votes.
yes+" "+"No:"+votes.no);
    }
    ;
    });
</script>
<script type="text/javascript">
function sendVote(vote) {
    $.ajax({
        method: "PUT",
        url:
        "/api/vote/"+vote
    }).done(function(data) {
        console.log("Got
back:"+JSON.stringify(data));
    });
}
</script>
<script type="text/babel">
class VoteButton extends
React.Component {
    constructor(props) {
        super(props);
    }
    render(props) {
        return (
```

Vote App (vote.html - React, Sockets, REST) (cont)

```
<button onClick=
{this.props.clickHandler} > {
this.props.label } </button>
);
}
}
class VoteComponent extends
React.Component {
constructor(props) {
super(props);
this.state = { }
this.yesClickHandler =
this.yesClickHandler.bind(this);
this.noClickHandler =
this.noClickHandler.bind(this);
}
yesClickHandler(e) {
sendVote("yes");
console.log("yes");
}
noClickHandler(e) {
sendVote("no");
console.log("no");
}
render() {
return (
<div style={{border:"1px solid
black"}} >
```

Vote App (vote.html - React, Sockets, REST) (cont)

```
{ this.props.proposition }
<VoteButton clickHandler=
{this.yesClickHandler}
label="yes" />
<VoteButton clickHandler=
{this.noClickHandler} label="no" />
</div>
);
}
}
ReactDOM.render( <VoteComponent
proposition="Want a poll?"/>,
document.getElementById("VOTE"));
</script>
</head>
```

Mongo

```
// CREATE
db.students.insert({ "id" :
1003675901, "name" : "Erich
Bachmann" })
// RETRIEVE
db.students.find( { "name":
"Erich Bachmann" } ).pretty() //
find student with name Erlich
Bachmann... pretty() makes easy to
read. .count() counts Erlichs
// other operators $gt, $lt, $eq,
$gte, ...
db.students.find( { "id" : { $gt :
"100" } } ) // students with id >
100
```

Mongo (cont)

```
// OR conditions
db.restaurants.find(
{ $or: [ { "cuisine":
"Italian" }, { "address.zipcode":
"10075" } ] }
)
// sort results
db.restaurants.find().sort( {
"borough": 1, "address.zipcode": 1
} ) // 1 = increasing, -1 =
decreasing
// UPDATE
db.restaurants.update(
{ "restaurant_id" : "41156888" },
{ $set: { "address.street": "East
31st Street" } }
)
// DELETE
db.restaurants.remove( {
"borough": "Manhattan" } ) //
remove all matching criteria
db.restaurants.remove( { } ) //
leaves the collection empty
db.restaurants.drop() // remove the
collection
```