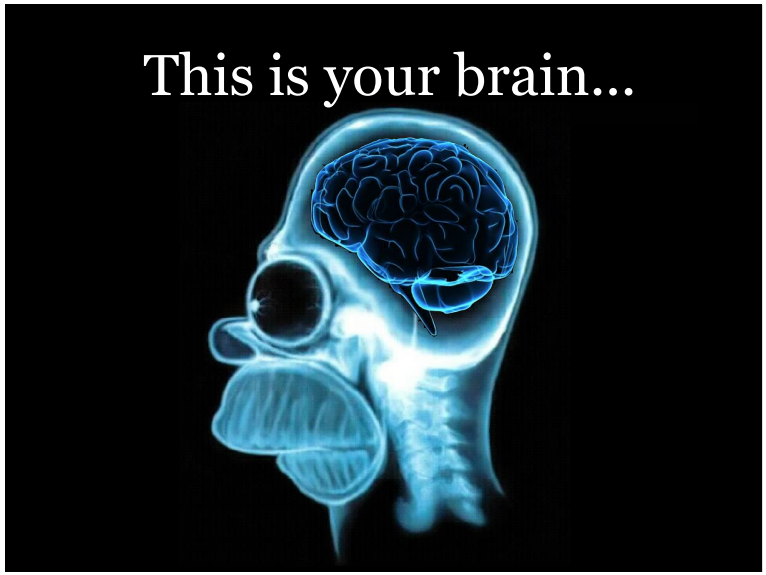
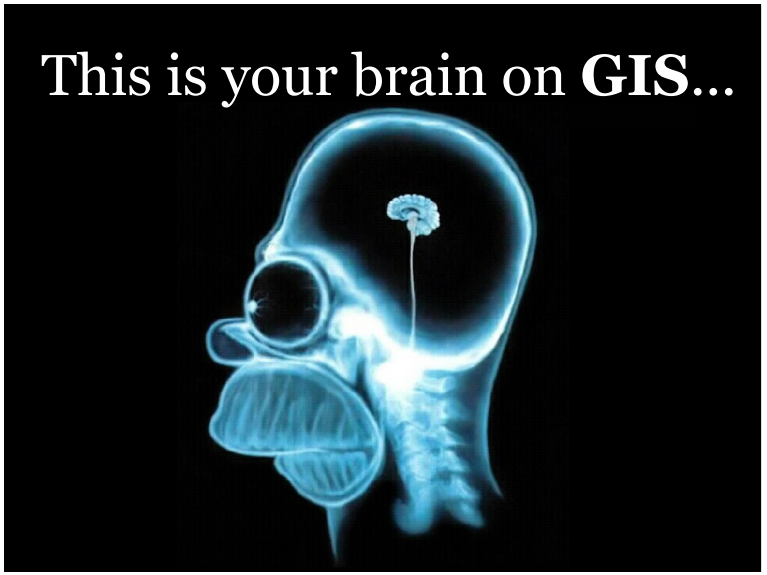
Web Services & Spatial SQL

Your Furry Funny Friends

I would like to start today make making the stakes as clear as I know how.
The stakes are high.
You are at risk.
It's important that you make the right choice, to grow up and have a healthy, happy life.

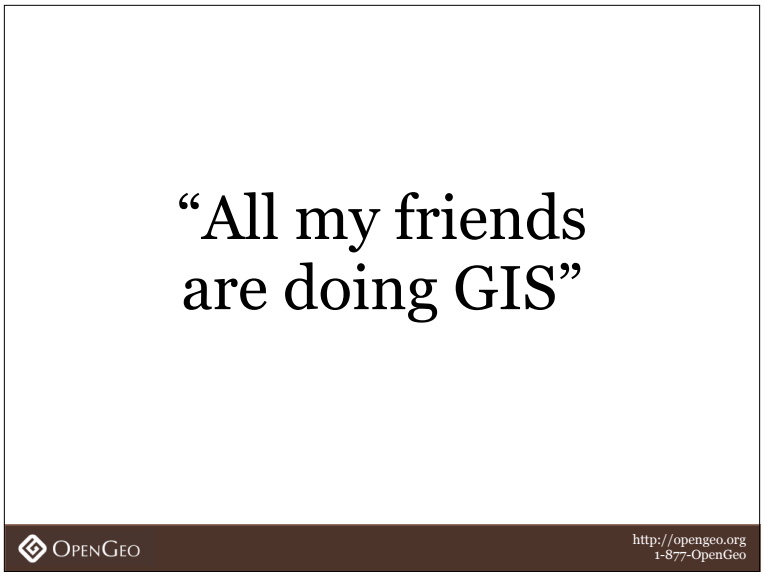


This is your brain.

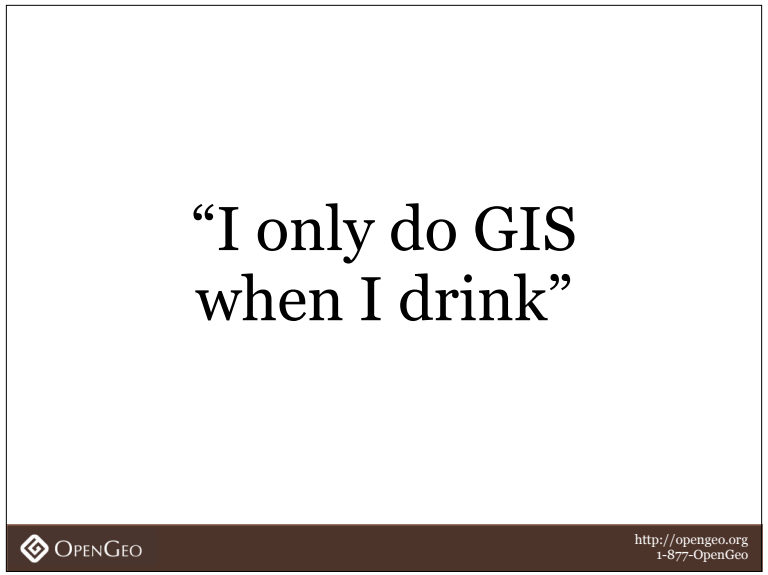


This is your brain on GIS.
Let me repeat myself just to be clear...
<repeat>
And... don't start with me...
I've seen too many lives ruined...
Heard the excuses too many times...

I can handle one hit of GIS.
Sure, you're a tough guy.

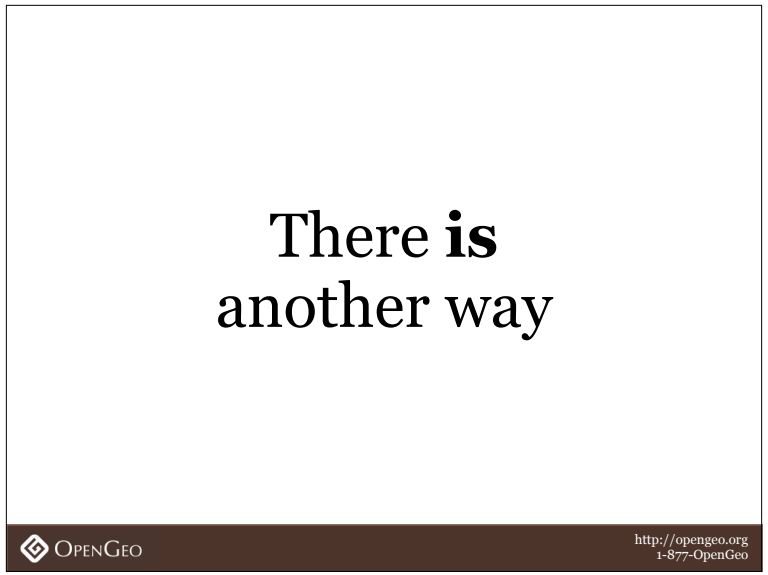


All of your friends are doing GIS.
And I bet if all your friends were jumping off a cliff, you'd do that too?

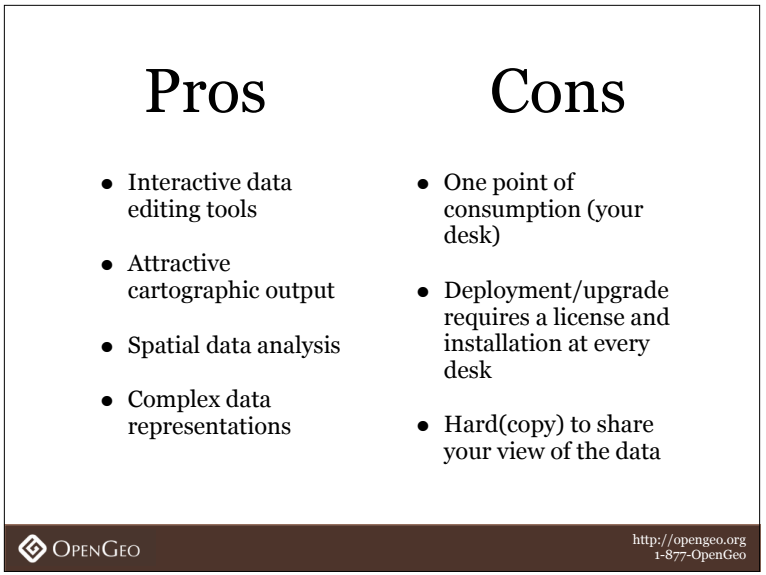


I only do GIS when I drink.
Right. That's how it starts.
Just one. Just with your friends. Just a few laughs over beers.
But then the years go by and one day you wake up and
and take a look at yourself in the mirror.

And there you are.
And look. It doesn't have to be this way.



There is another way.
Let's look at why people do GIS.
I'm a recovering GIS user myself. I understand.



There's a lot of positive reasons people do GIS
...pros.
But there's substantial downsides too, mostly due to the fact that GIS sits on your desk.
...cons...
It's a big, complex lump of software and it trains us to think

that GIS is the one tool to rule them all.

Even on the desktop we use different tools
for graphic design (like InDesign),
and data query (like Access)
and data analysis (like Excel).

Why do we only use one tool when we're working with maps?
It's time to break free.

Spatial IT

- Store and query spatial information
- Analyze spatial information
- Symbolize spatial information
- Share spatial information

We don't do GIS. We're information technology experts, who understand spatial problems.

We ...<bullets>...

We do spatial IT.
And here's the punchline. Finally.

On the web

We should do spatial IT on the web.
Why should we do spatial IT on the web? Let's look at the pros and cons of GIS.

Pros

- Interac… editing tools
- Attrac… carto…
- Spati… analysis
- Com… data rep…

we can do this on the web!

Cons

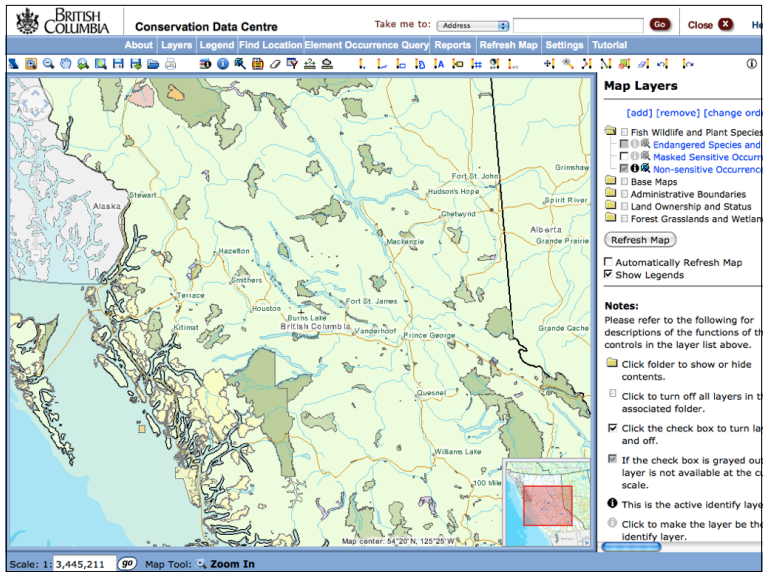
- One point of consumption (your desk)
- Deployment requires a license installation at every desk
- Hard(copy) to share your view of the data

not a problem on the web!

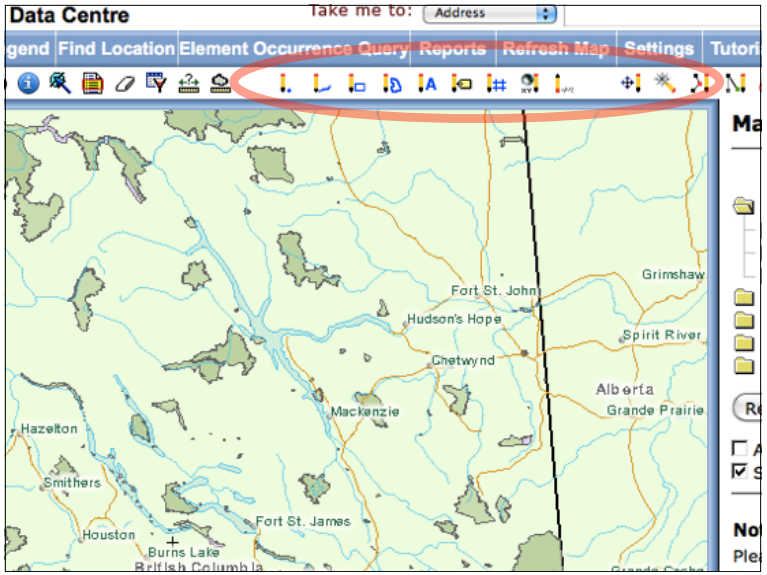
Web technology, and it pains me to say this as a server guy,
web technology can now do almost everything we used to do on the desktop.
It can do it cheaper, it can do it more flexibly.
...Pros, Tim, ..., Paul, Martin...

And when you're on the web, all the cons of the desktop disappear.
...Cons...

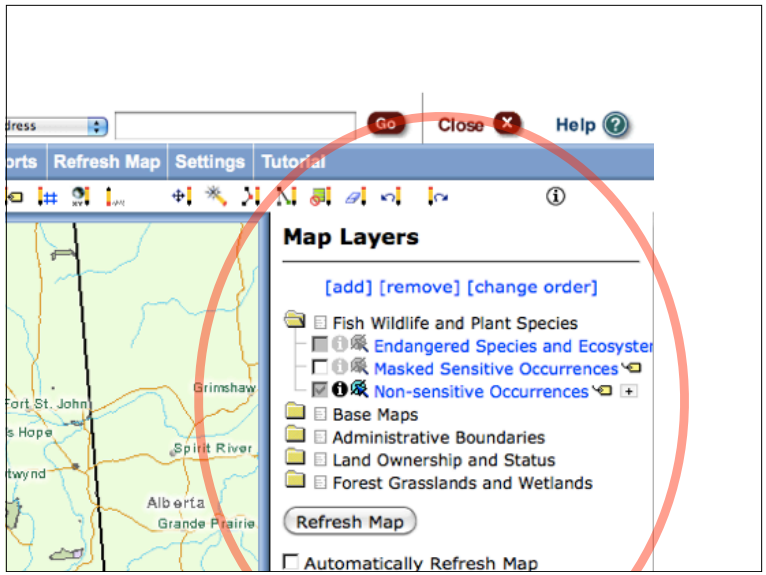
And I'm not talking about putting GIS onto the web,
Just translating the one ring of desktop GIS over to web technologies



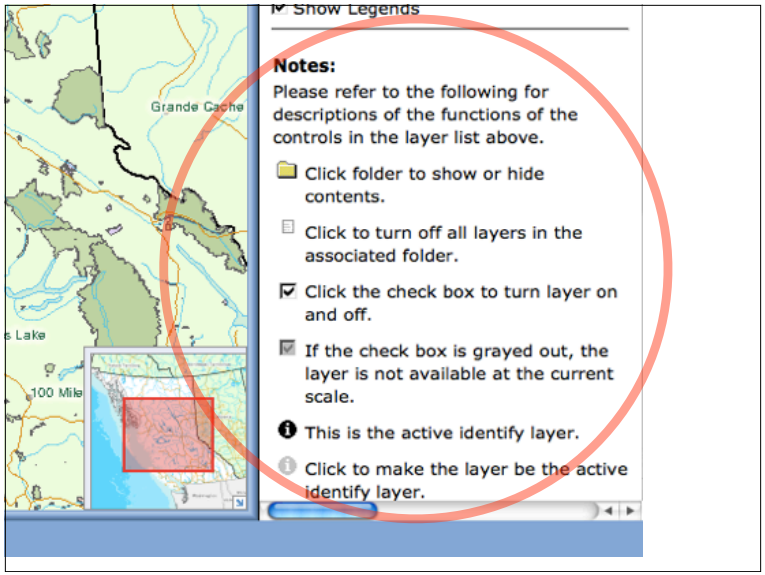
like this super site
from my home province of British Columbia



With the classic row of fancy tools that only GIS people understand



The list of layers, every possible layer,
the “active layer” concept, borrowed from the desktop experience
(because everyone knows what an “active layer” is
(it’s the one with the dark “i”, by the way))



and then the need to embed directions right into the user interface,
because it’s so complex that normal folks can’t figure it out on their own.
This is not the way.
There is a truth about web mapping applications we need to examine...

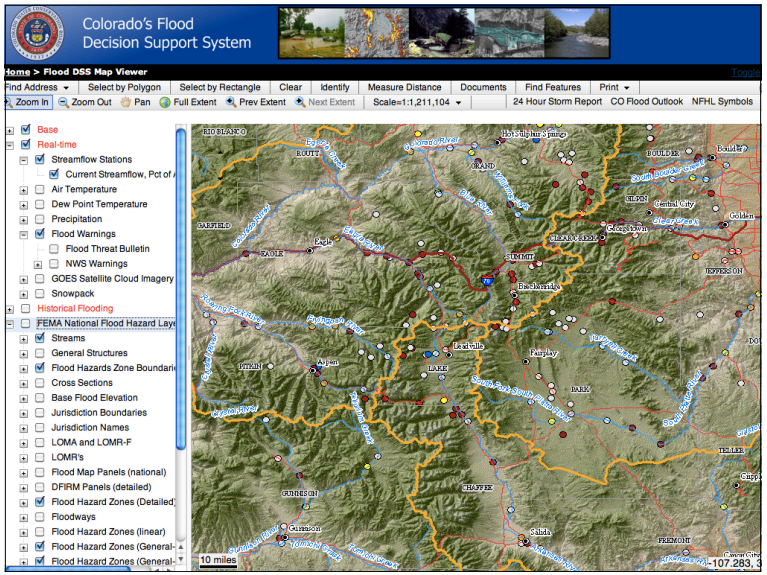
The Truth

The truth came knocking at my door, and I said,
"Go away! I'm looking for the truth!"
And so it did.

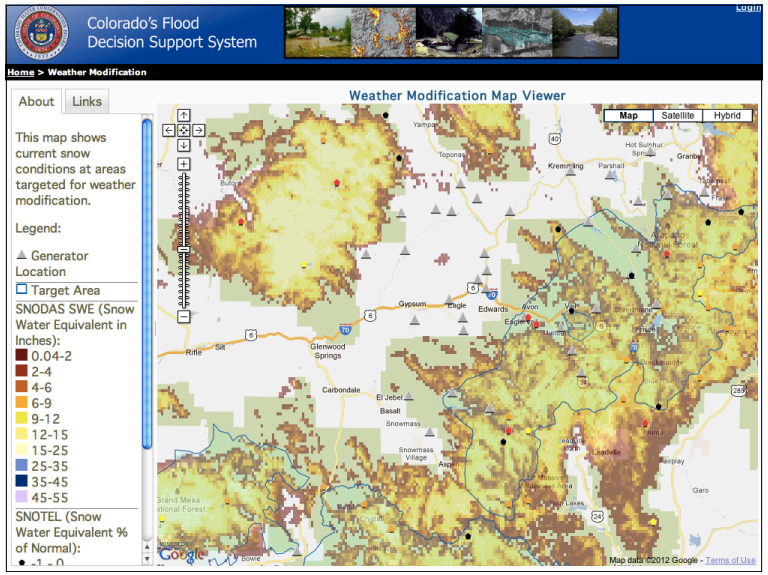
- Robert Pirsig, Zen and the Art of Motorcycle Maintenance

...read....

The daily quotidian affairs and apparent requirements of our users,
what they *say* they need,
as opposed to what they *do* need,
can get in the way of our pursuit of the truth.



My colleague Ian Schneider build this app back when he was a consultant.
It's a "decision support system".
Which is to say, it doesn't have a single purpose, it's built in the hope of **finding** a purpose.
He didn't build this site malevolently,
he carefully built exactly what the client requested, Feature by feature.
In theory it serves everyone, in practice it serves no one.



But when the pressure to deliver abated,
and Ian had some time to contemplate, the truth came.
A single "decision support application" cannot meet every purpose,
but on the web, it's possible for each purpose
to have their **own** decision support application...

This example stripped down the original app to
just the basemap and a handful of layers of realtime data
that weather managers need to make decisions about when to do cloud seeding.
It truly supports decisions, but only one kind of decision.
Note: They can't turn layers on and off. They don't need to.
There's no instruction panel. They don't need one.

The Truth

Every application has
one purpose.

OpenGeo

http://opengeo.org
1-877-OpenGeo

And so we come to the simple truth.
Every (good) application has one purpose.

The Truth

If you can't name that purpose,
don't build the app.

And if you can't name that purpose, you shouldn't build the application.
I can go even further down this road, and state that

The Truth

The best applications have only two layers.

A base map. And a layer of interest.

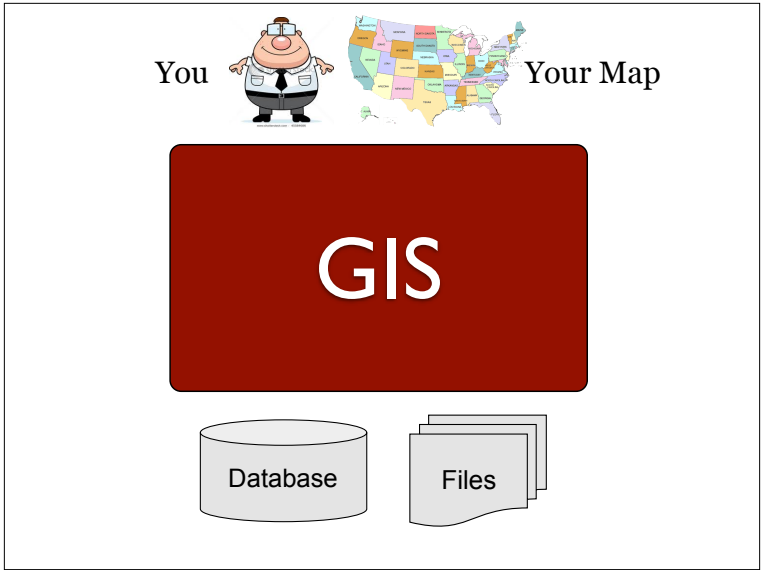
the best spatial applications have only two layers a base map, and a layer of interest
And here's the best part, here's the part that blows my mind

The Truth

They don't have to come
from the same place.

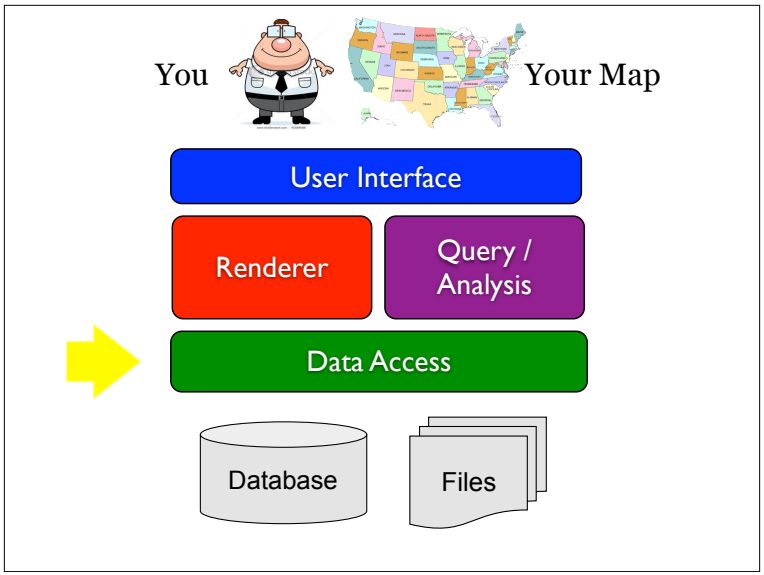
the layers don't even have to come from the same place.

A good web application may appear on your screen as a
single, composed piece.
A pretty map, a useful tool, an analytical display.
But each component can be served from a different location,
from a different server, from a different organization entirely!



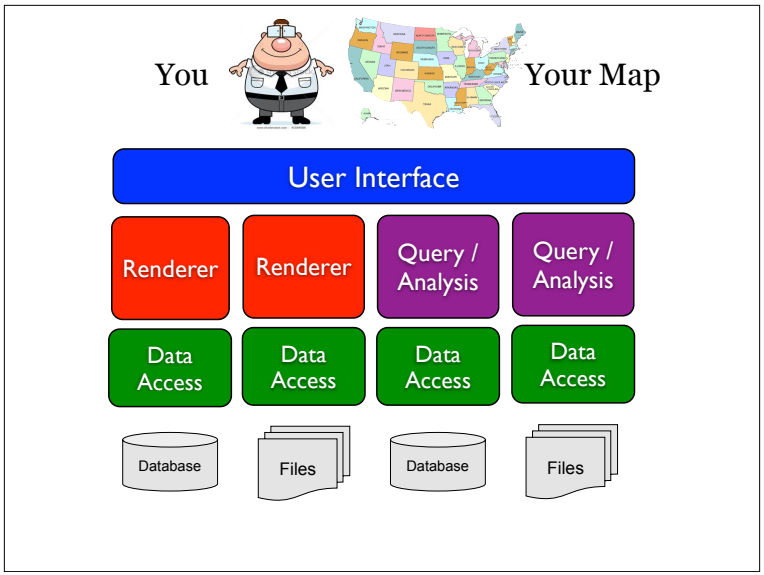
We have gotten used to thinking about "GIS" as a big functional blob,
a single piece of functionality that mediates our relationship with data in databases and files.

But it's not.
It's a collection of functions.

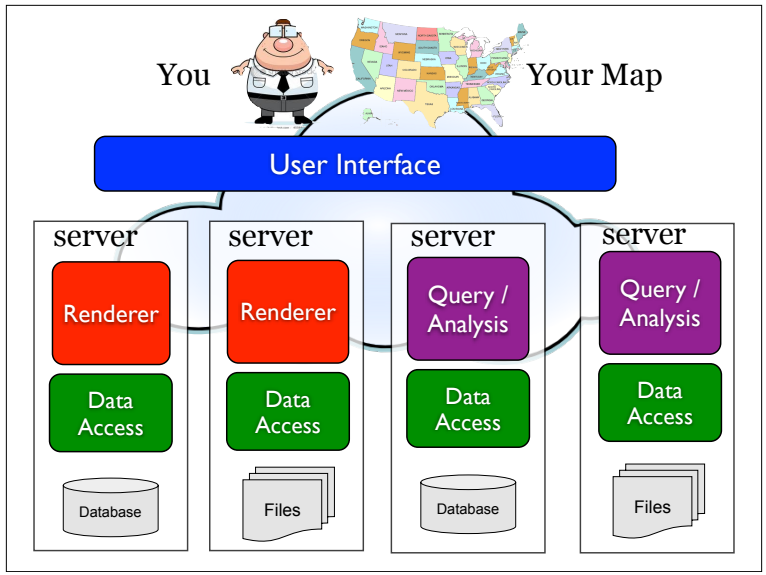


A GIS is
<X> a data access layer to abstract different formats and databases,
<X> A rendering layer to turn the raw data into cartographic output
<X> A query and analysis layer to extract pieces of the data or transform them
and
<X> a user interface to allow the user to manipulate the data, the styling, the queries and the analyses.

And here's the crazy part. Those functions don't have to run on the same machine the user sits at. They don't even have to run on the same continent.



On the web, each function is separable, and an application can bind multiple functions together
into one interface.

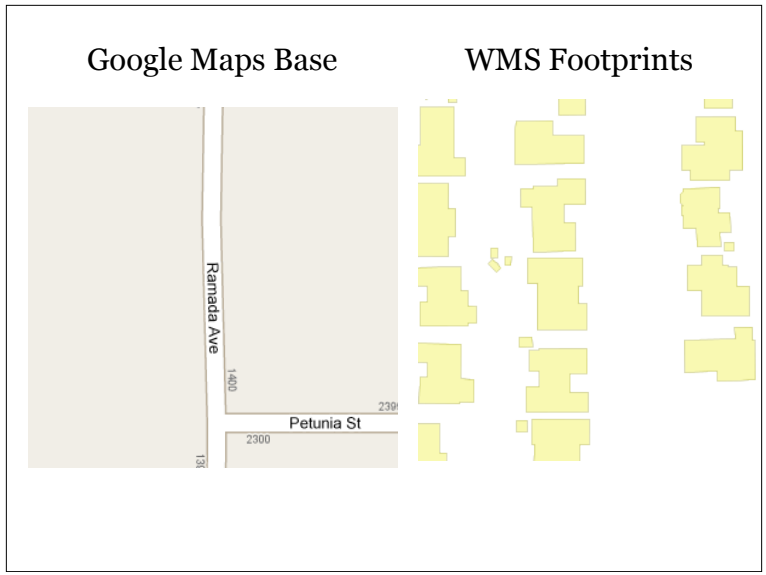


And each of those functions can run on different servers,
and each server can be run by a different organization.

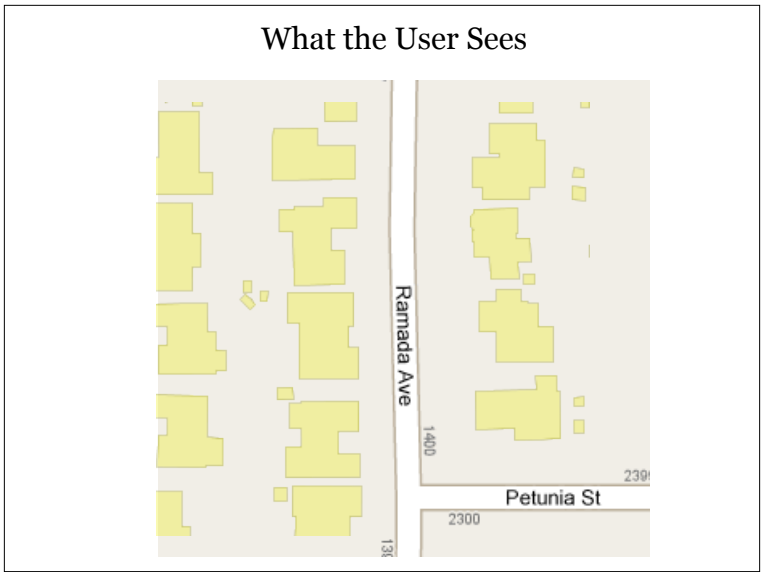
Simple Example

- Basemap +
- Web Map Service (WMS)

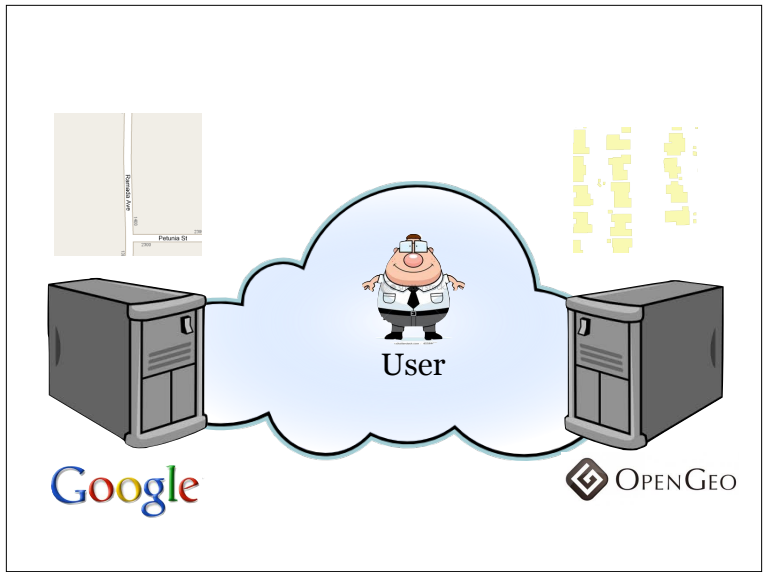
Here's the simplest example. A two layer map.
A basemap to provide context, and a layer of interest drawn using a web map service, a remote rendering service.



Google can provide a base map.
And the remote renderer can provide the overlay.
All that is necessary to synchronize the result is to ensure that both layers are pulled in using the same projection and scale.



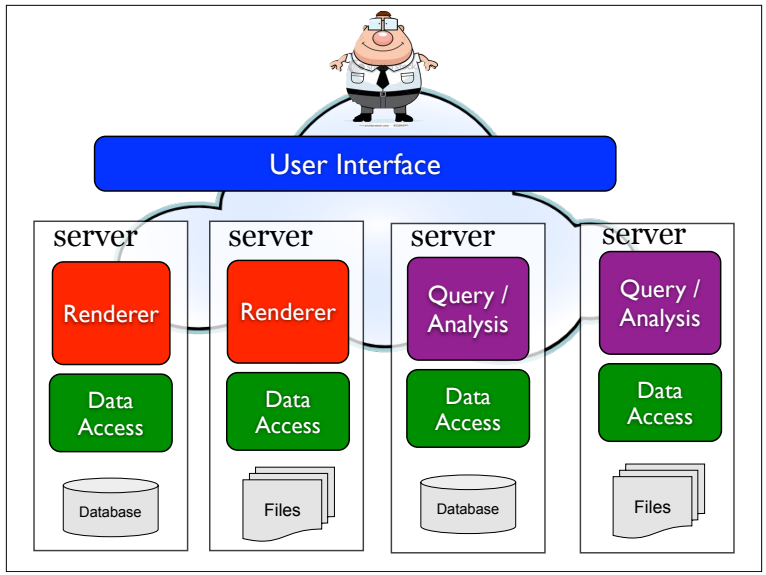
And the result can be composited in the users web browser.



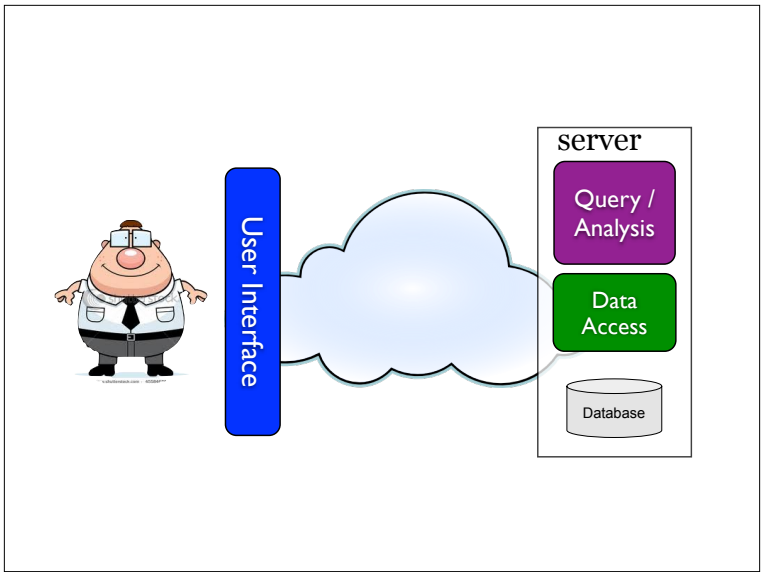
The user doesn't need to know or care that the map he is seeing is actually produced in two separate places,
from two separate sources of data,
from two completely different organizations in fact.

Ignorance is Bliss

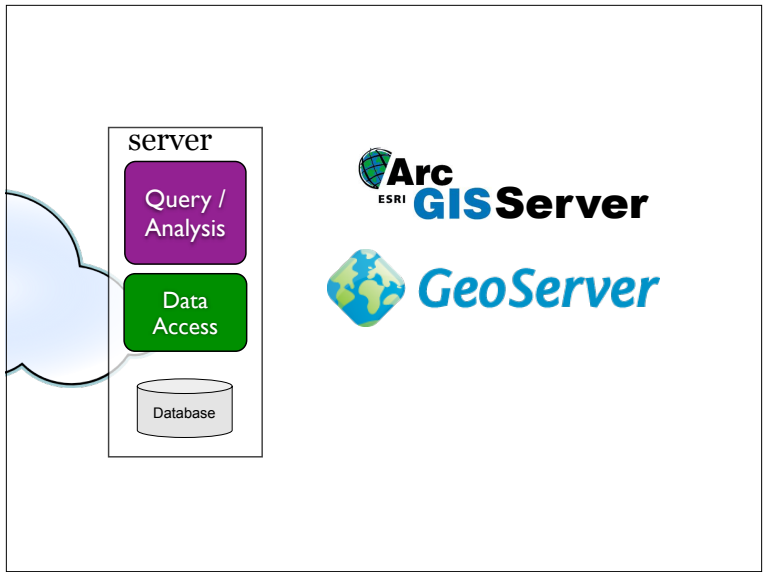
The user can just get his job done.



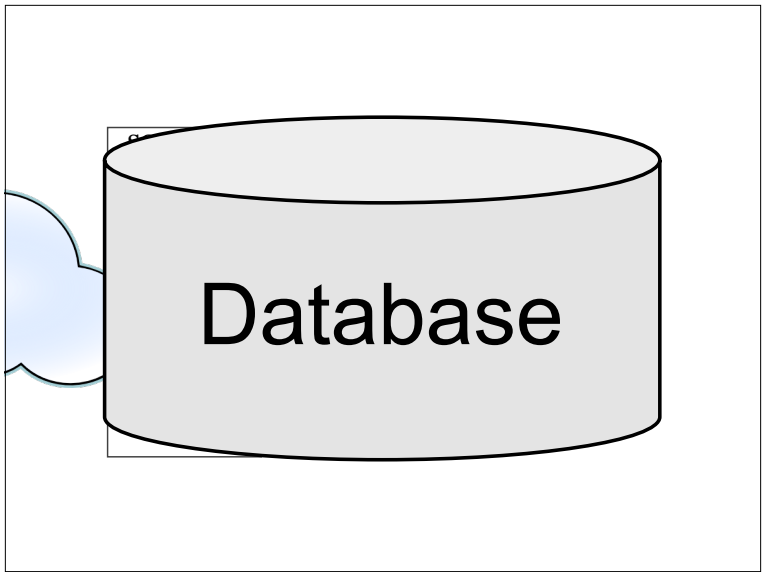
And here's where we get to the part that really gets me excited.
In addition to breaking the rendering function into distributed parts,
web architectures also let us distribute the analysis function.



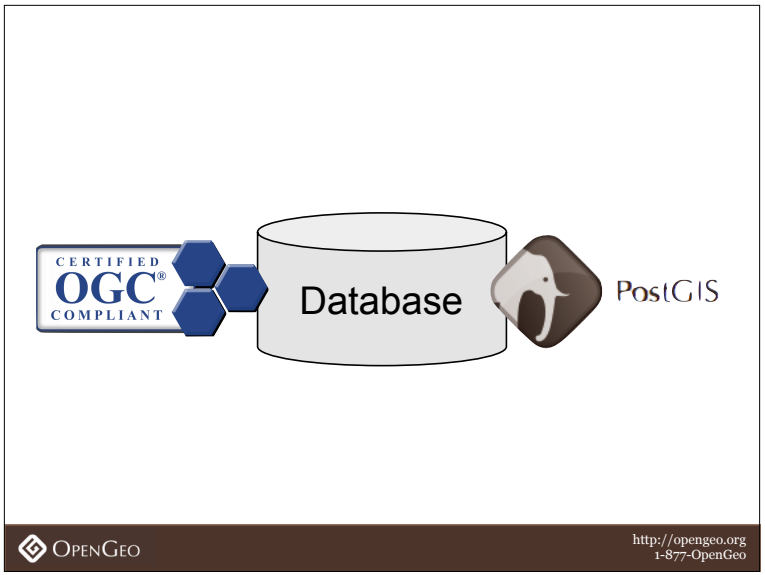
So the user can pull analytical results from remote servers.
And this is where I start to quiver a little,
this is the really good stuff,
because when I say "analytical services"



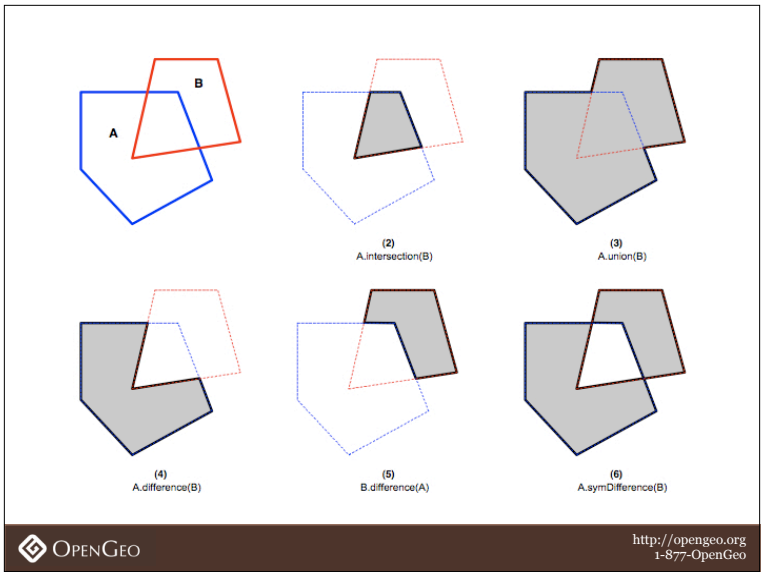
that doesn't mean you need to deploy ArcGIS Server or
GeoServer (sorry Justin)
or use fancy protocols like WFS or WPS or whatever ESRI calls their geoprocessing stuff,
you can get enough spatial analysis to solve 90% of your problems using



just a spatial database
just a big grey cylinder
I mean, the savings on architecture diagrams alone, are potentially huge



An Open Geospatial Consortium Simple Features for SQL database,
like PostGIS has all the functionality we need.



It has all the spatial tests and operations you find in middleware like GeoServer,
and more important,
it has the ability to evaluate complex data processing and summary queries.

...read...

That sounds like an analytical question!
How many lines of code should it take to solve it?

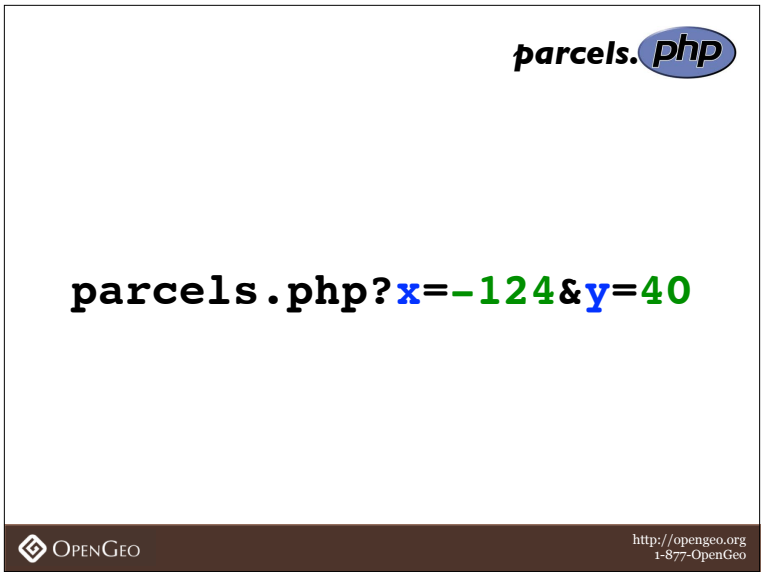
One line!
Using one spatial function, a coordinate, and a table of parcel data, we can generate a classic GIS "alert list" of people to phone about the fire.

Web Services?!?

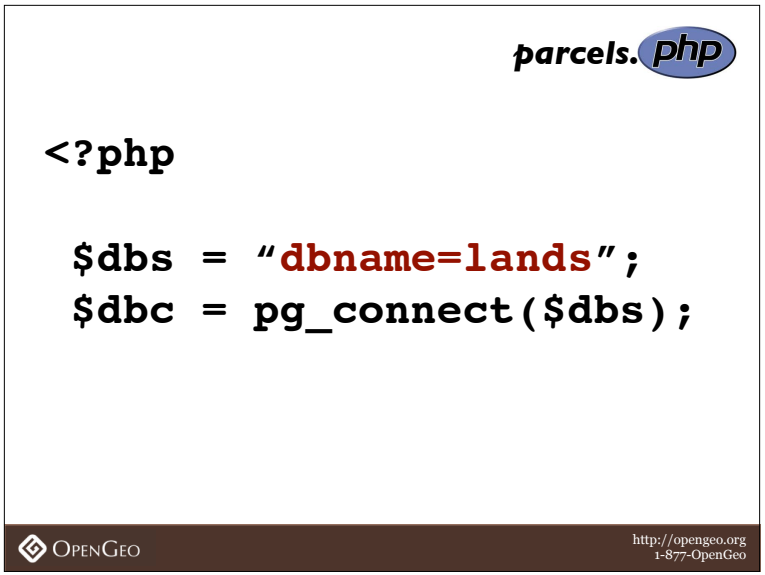
But wait a second, I've been talking about integrating services over the web... how do we do this SQL query over the web?
The answer, thin thin thin little scripts.


```
<?php
  $var = stuff();
?>
```


Here's an example of a web service script in PHP that wraps a SQL query and returns GeoJSON.
Why php? I like it! (everyone else hates it)
but it has $variables and semi-colons;
so it reminds me simultaneously of perl and C
and that makes me comfortable.



So, a simple script, it takes in URL parameters, things that area easy to generate from a click on a web page
and spits back a JSON answer.



First, connect to the database.

```
$sql =
 'SELECT
    ST_AsGeoJSON(geog),
    owner_phone
  FROM parcels WHERE
   ST_DWithin(
    geog,
    ST_MakePoint($1,$2),
    1000
   )';
```

parcels.php

OpenGeo http://opengeo.org 1-877-OpenGeo

Second, here's the SQL we are going to run. It's our example phone-list query for a point of interest.
Note that we're going to return GeoJSON, and the database supports output in that format!

```
$dbh = pg_prepare(
          $dbc,"pqry",
          $sql);

$rslt = pg_execute(
           $dbc,"pqry"
           $x,$y);
```

parcels.php

OpenGeo http://opengeo.org 1-877-OpenGeo

We create a prepared statement using the SQL. This gives us some protection against SQL injection attacks.
Then we execute the query, passing in the X and Y values we got from the HTTP call.

parcels.php

```
header("Content-type:
        application/json");

print '
{ "type": "FeatureCollection",
  "features": [';
```


Now, we get ready to return the answer! Send the content type (JSON) and start the JSON feature collection.

parcels.php

```
while($r = pg_fetch_row($rslt))
{
  ....
}
```

OpenGeo http://opengeo.org 1-877-OpenGeo

For each feature in the result set we...

parcels.php

```
if ($i++) print ",";

printf('
 { "type":"Feature",
   "geometry":%s,
   "properties":{"phone":"%s"} }',
 $r[0], $r[1]);
```

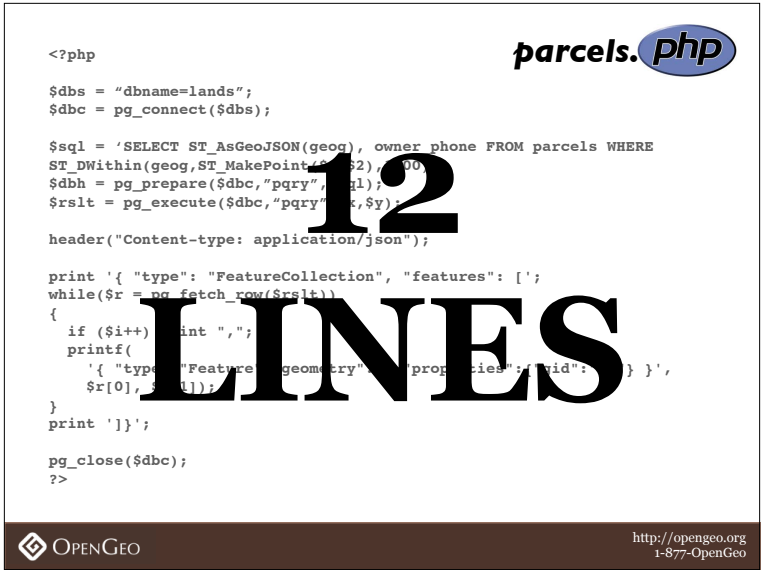

put a comma between each feature
print a GeoJSON feature, inserting
the GeoJSON geometry result into the geometry property and
the phone number into the non-spatial properties

parcels.php

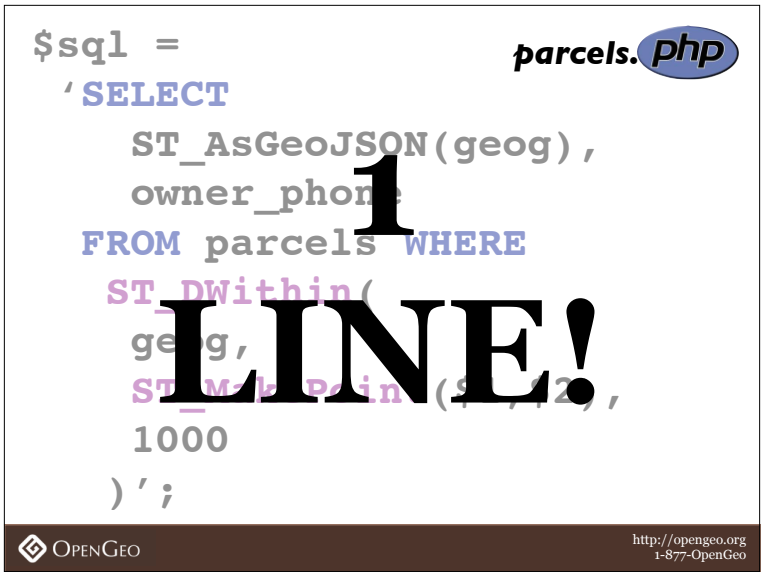
```
print ']}';

pg_close($dbc);

?>
```


when the loop completes,
we close out the jSON feature shut down the database connection and exit.
OK, hey, maybe you're thinking
"that was actually not simple but kind of complex",
and I'm a generous and empathetic guy,
but frankly,
NO, YOU'RE WRONG

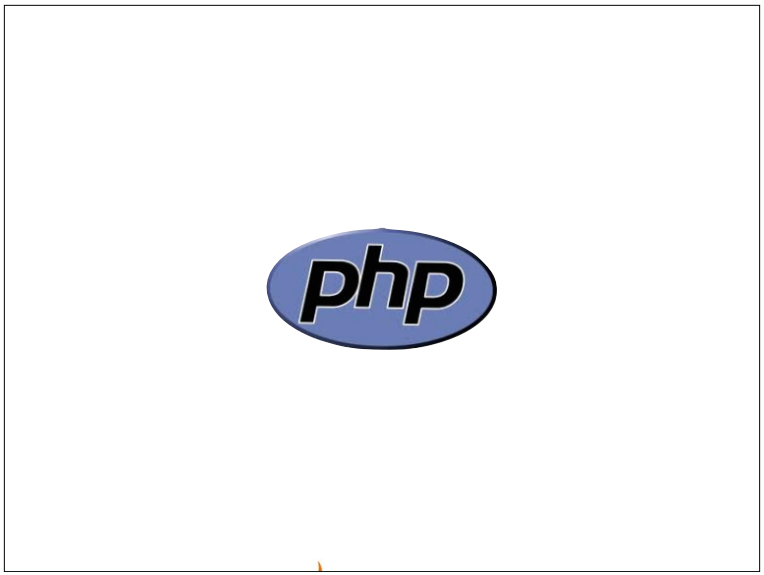
This is the whole thing on one page and it’s very small.
<X> You connect.
<X> You run some SQL.
<X> You return the result in the format of your choice.
<X> You disconnect.



Our PHP example is just 12 lines long!
But really, those are just 12 lines of boilerplate,

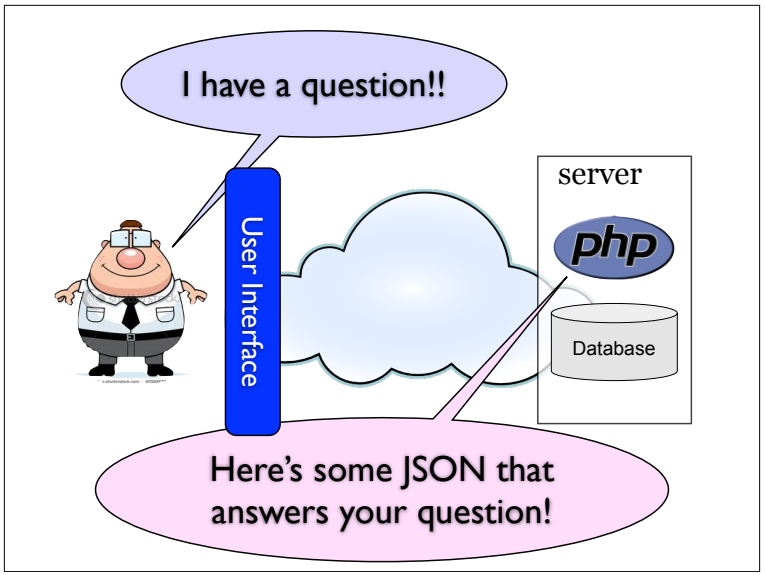


the logic of the thing is bound into just one SQL statement!
And this is a really simple one. We can do so much more!



And please, don't get hung up on the fact that my example used PHP!
I like PHP, but you can write simple database wrapping web service scripts in **any language at all.**

Scripting languages like perl, ruby, python and ASP.
Even full languages like Java, .C# and C++.
The important point here is that it's
easy to tie your database to a web application,
and make use of all the logic inside that database.



This is fabulous.
We have an architecture that allows
a web browser application to call into a server
with a simple parameterized query and
get back a result it can directly process using JavasScript.
The amount of software you need to deploy is **minimal**,
the system complexity is **low**, the functional return is **high**.
Yeah!**

So?

“How many people live with 1km of <click>?”

OK, you’re not impressed.
Here’s some questions I could answer in
one line of SQL,
and which therefore could be potentially bound into
a spatial web application using this technique:

“What’s the average elevation of <draws polygon>?”

“How far away from me is the nearest <something>?”

school
well
base
helicopter
thai restaurant

"It's time to pray, what direction is Mecca?"

"What's the average home price in this neighborhood?"

start thinking mobile phones,
you don't even have to click to make this work

"When is the next bus going to be here?"

"Where's that plane going?"

"Where can a guy get a drink around here?"


```
$sql =
 'SELECT
    ST_AsGeoJSON(geog),
    owner_phone
  FROM parcels WHERE
   ST_DWithin(
    geog,
    ST_MakePoint($1,$2),
    1000
   )';
```

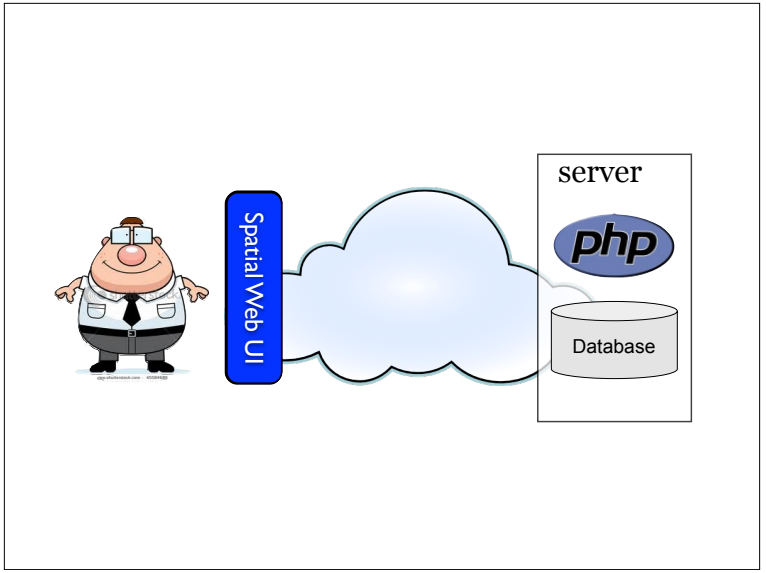
1 LINE!

Given the right data tables, these questions can all be answered in one line of SQL.

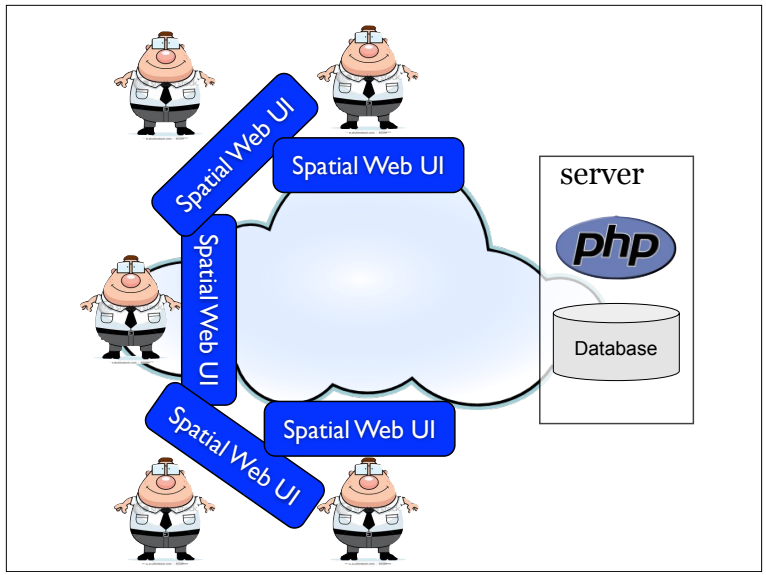
```
<?php

$dbs = “dbname=lands”;
$dbc = pg_connect($dbs);

$sql = ‘SELECT ST_AsGeoJSON(geog), owner, phone FROM parcels WHERE
ST_DWithin(geog,ST_MakePoint($1,$2),500)’;
$dbh = pg_prepare($dbc,”pqry”,$sql);
$rslt = pg_execute($dbc,“pqry”,array($x,$y));

header("Content-type: application/json");

print '{ "type": "FeatureCollection", "features": [';
while($r = pg_fetch_row($rslt))
{
  if ($i++) print ",";
  printf(
    '{ "type": "Feature", "geometry": %s, "properties": {"id": %d} }',
    $r[0], $r[1]);
}
print ']}';

pg_close($dbc);
?>
```

12 LINES

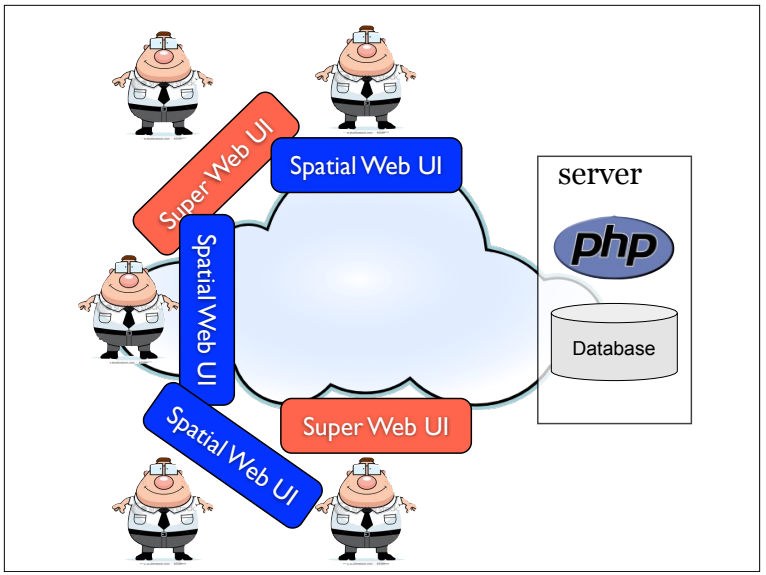
which means they can be exposed onto the web with just
12 lines of scripting glue



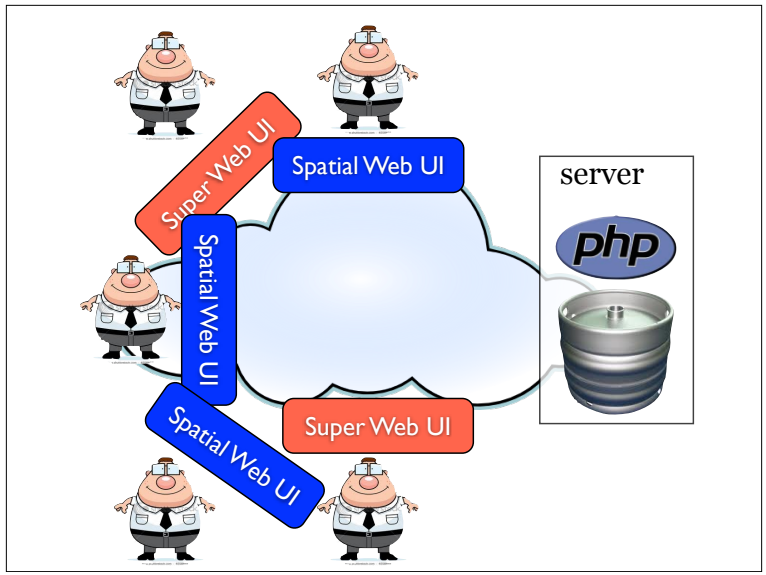
Which means you can provide that service and answer those complex spatial questions for any spatial web application in your organization, or in the world.



And you can provide the service to multiple users without **any extra software deployment**.



And that same service can actually feed different applications,
because it's just one component among many.



And if you get a new database with better data, you can upgrade the service to **all your users simultaneously**.

Wow

OK, so I assume you all want one of these now... it's amazing, it slices, it dices, it plays the piano.

The next few talks are going to show some web services in action,
and when you see them, remember that
what you're seeing is just ONE INSTANCE of how the service can be deployed.
Each service can be re-mixed into different interfaces and different contexts.

As professionals, I think it's time we re-thought how we approach the job of doing spatial and map work.

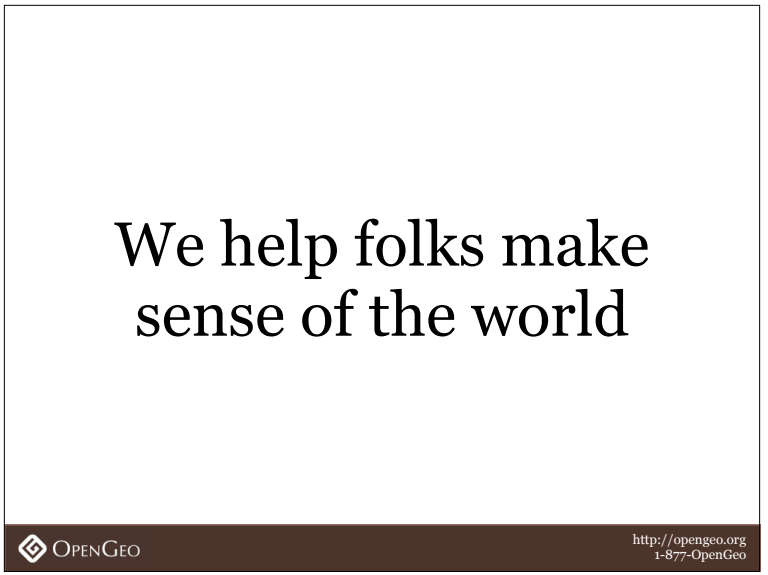
We don’t do GIS. That’s not our job.

We query data.

We visualize patterns.

We make maps.

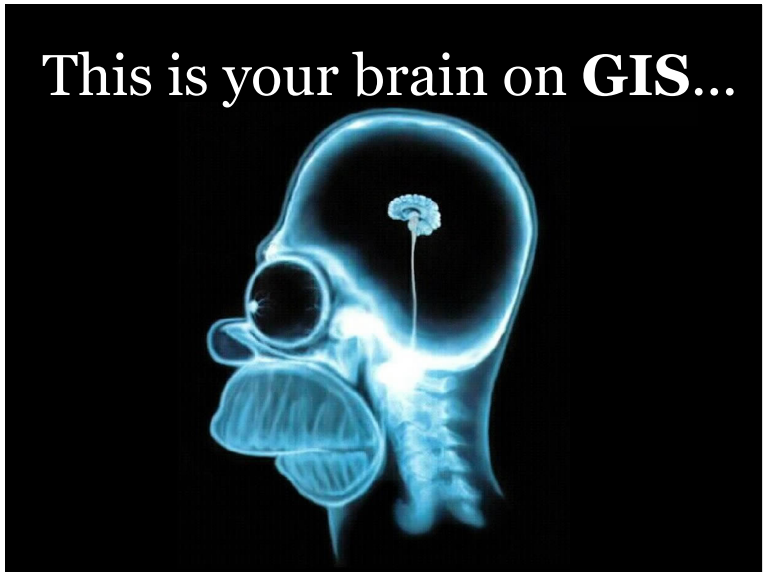
We share our findings.



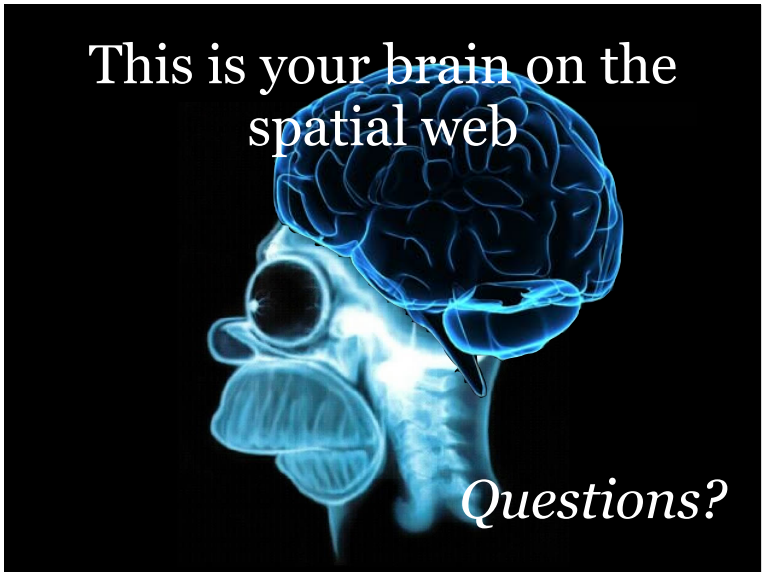
We help folks make sense of the world.

We don’t do GIS. That’s not our job.

We do spatial IT,
on the spatial web.



This is your brain on GIS.



This is your brain on the spatial web.
<X> Thanks. Any questions?