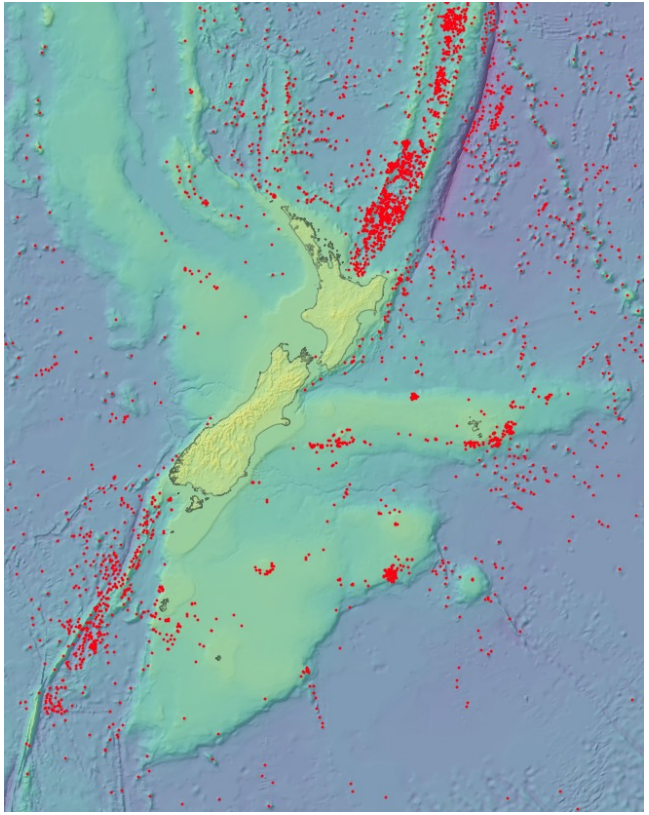
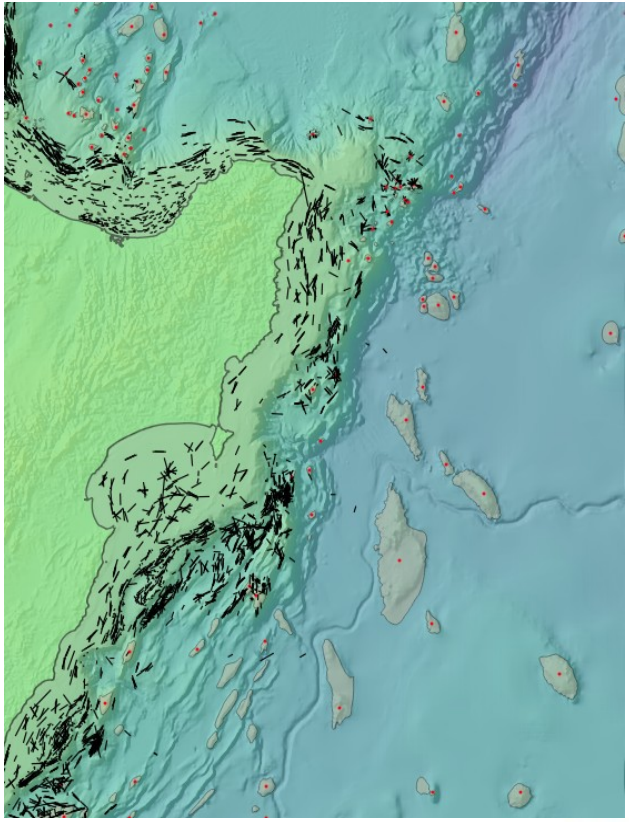
New Zealand Seamount Database

Brent Wood

NIWA





A story about linked data, but not as W3C defines it

National Institute of Water and Atmospheric Research (NIWA)

New Zealand Crown Research Institute (CRI)

Government owned commercially funded environmental research institute, with science centres focused on:

Aquaculture
Atmosphere
Climate
Coasts and Oceans
Environmental Information
Fisheries
Freshwater and Estuaries
Natural Hazards
Pacific Rim

http://niwa.co.nz

NIWA
Taihoro Nukurangi

Who am I?

Brent Wood

Joined NIWA (or its predecessor) in 1975

Fisheries field/seagoing technician

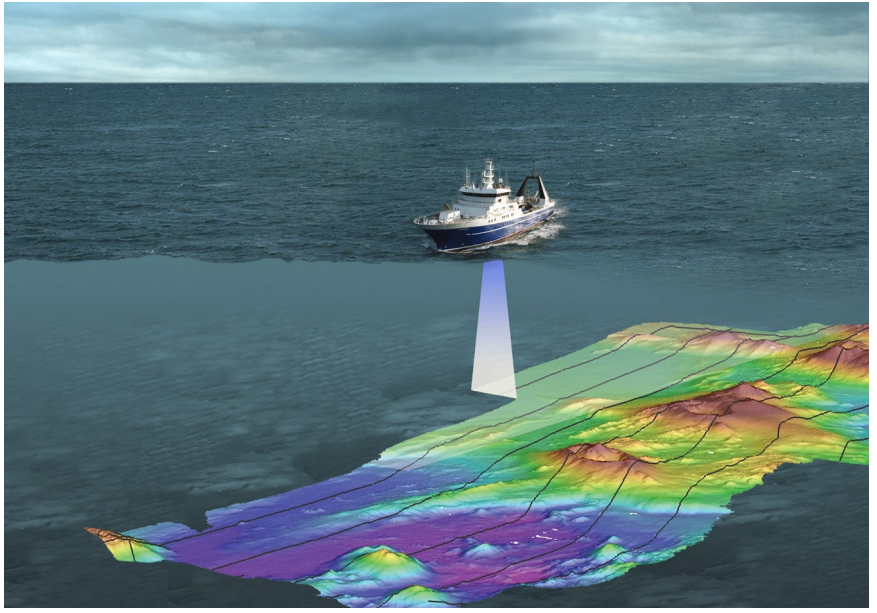
Data manager/database manager/database designer

Metadata manager (Geonetwork)

Open source GIS user (QGIS, PostGIS, Spatialite, GMT, Mapserver, Geoserver, R, ...)

NIWA
Taihoro Nukurangi

Seamounts



”Oases of the ocean”

Defined as > 1000m elevation,
but we include knolls & hills

About 3000 around New Zealand

Understanding is critical for managing fishing, mining, etc

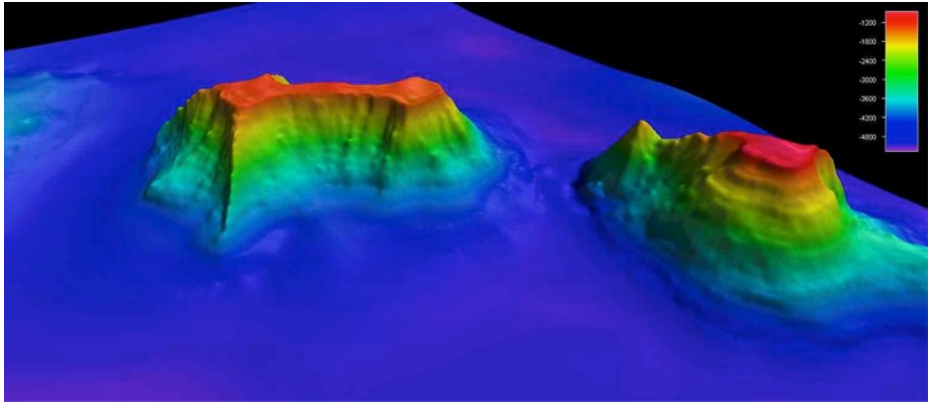
The project

Funded by Fisheries New Zealand, project BEN2020/07

Used NIWA, GEBCO and fishing industry data

Included an assessment of commercial fishing effort on seamounts

Used GIS analyses to identify and define underwater features



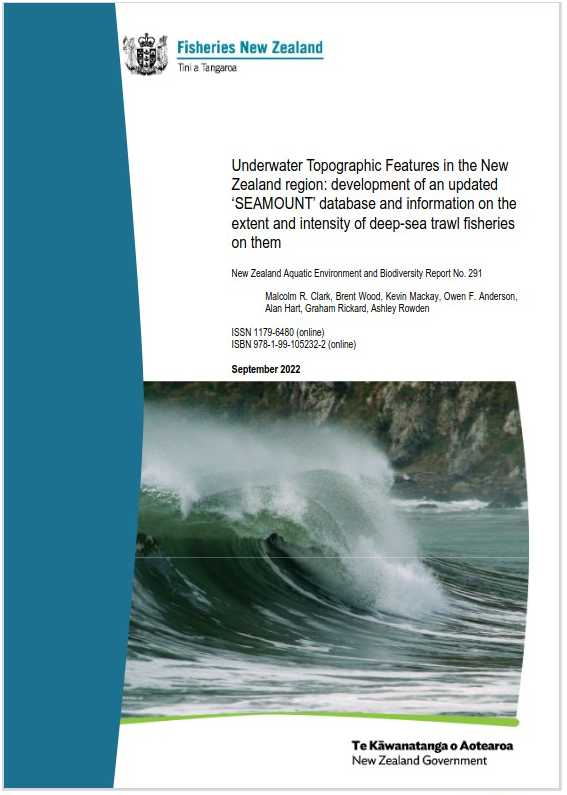
The database

Describes 2964 Underwater Topographic Features (UTF's)

414 seamounts (>= 1000m elevation)
1495 knolls (250 – 999 m)
1055 hills (100 –249 m)

NZ Territorial Sea (89)
NZ EEZ (1907)
outside the EEZ (968)

Includes 43 columns for each feature including physical, oceanographic, geological, biological and fisheries related descriptors



NIWA
Taihoro Nukurangi

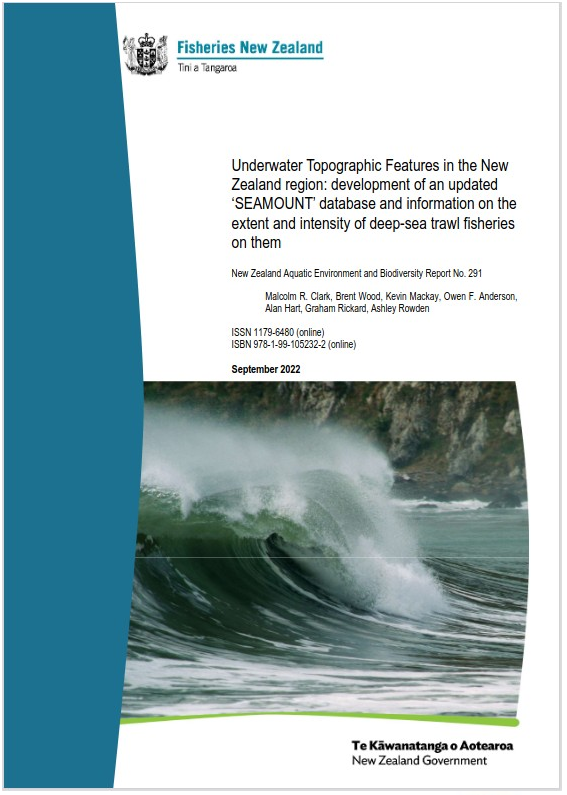
The database

Describes 2964 Underwater Topographic Features (UTF's)

414 seamounts (>= 1000m elevation)
1495 knolls (250 – 999 m)
1055 hills (100 –249 m)

NZ Territorial Sea (89)
NZ EEZ (1907)
outside the EEZ (968)

Includes 43 columns for each feature including physical, oceanographic, geological, biological and fisheries related descriptors



NIWA
Taihoro Nukurangi

The database

That is business as usual for a database, and not what this is about!

What is this database typically used for?

What makes it different to many other databases?

The database

What is this database typically used for?

To answer questions about what is known about a seamount

For simple questions like

- how high?
- what area does it cover?
- how steep is it?
- where is it?
- which water mass is it under?
- how was it created?

The answers come from information in this database

NIWA
Taihoro Nukurangi

The database

More useful questions relate to information NOT in this database

What National Invertebrate Collection (NIC) specimens are there from a UTF?

What research trawls (and catches) took place on this UTF?

What commercial fishing (and catches) took place on this UTF?

These sorts of questions require extracts from multiple databases, to be merged and analysed somewhere to determine the answer

There is a better way!

NIWA
Taihoro Nukurangi

The database

Is implemented using the PostgreSQL ORDBMS database application

It uses the PostGIS extension to support spatial data
(points, lines and polygons)

It uses the Postgres Foreign Data Wrapper capability to directly link to external databases, I will present two examples:

The research trawl database is another PostgreSQL/PostGIS database (but completely separate)

The NIC uses a non-spatial MySQL database (not Postgres)

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

These are installed in a Postgres database as extensions just like PostGIS

They allow foreign tables to be created in a Postgres database - these do not store data, but are pointers to data stored elsewhere

There are several different FDW extensions available, we need two for this use case

postgres_fdw	create foreign tables which are links to tables in remote Postgres databases
ogr_fdw	create foreign tables which are links to tables in any OGR compatible data source

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

This presentation discusses the use of Foreign Data Wrappers to allow simple SQL's run in the seamount database to directly query

the remote trawl database (station and catch data)

the MySQL NIC database (museum specimen data)

as if the remote data is stored in local database tables

This functionality replaces hours or days of data extracts and analysis with a local SQL query returning an answer in a few seconds

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

The research trawl foreign tables

We need access to the research trawl:

station data (where, when, etc)
catch data (what species, how much)

So we will create two foreign tables – station and catch

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

install the postgres_fdw extension

```
create extension postgres_fdw;
```

create the trawl fdw server

```
create server trawl_fdw
        foreign data wrapper postgres_fdw
        options (host 'servername', port '5432', dbname 'dbname');
```

create user mapping for fdw

```
create user mapping for user
    server trawl_fdw
    options (user 'username', password 'password');
```

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

mapping external trawl tables as local foreign tables

import foreign schema trawl
limit to (trawl.t_station, trawl.t_catch)
from server trawl_fdw
into public;"

this command allows ALL the tables in the remote schema to be set up as foreign tables – we limit it to just the two we want

list the foreign tables now present in our database:

Schema	Name	Type	Owner
public	t_catch	foreign table	
public	t_station	foreign table	

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

We can now query our foreign table – Postgres will retrieve the data from the remote database – eg: how many trips are there?

```
select count(distinct trip_code) from t_station;
 count
-------
   719
```

And how many species recorded from all surveys?

```
select count(distinct species) from t_catch;
 count
-------
  1443
```

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

And we can (spatially) join local seamount data to trawl data in SQL queries

```
select s.reg_no,
       s.name,
       st.trip_code,
       st.station_no,
       c.species,
       coalesce(c.weight, 0.100) as weight – set any null weights to a nominal 100g
from t_station st,
       t_catch c,
       seamount s,
       seamount_polygons p
where ST_Intersects(ST_Transform(st.track, 3994),p.poly)
  and p.seamount = s.reg_no
  and st.trip_code||'_'|| st.station_no = c.trip_code ||'_'|| c.station_no;
```

reg_no	name	trip_code	station_no	species	weight
654	The Pimple	aex0101	19	SSO	11263.300
654	The Pimple	aex0101	19	BEE	1.200
654	The Pimple	aex0101	19	BOE	289.400
657	Hegerville	aex0101	27	CSU	1.900

Foreign Data Wrappers

Querying remote Postgres databases from a Postgres database is a reasonable thing to expect.

Now we look at the ogr_fdw extension.

This allows ANY accessible OGR datatype to be set up as a foreign table in a Postgres database:

> Shapefiles, WFS services, ODBC data sources, Mapinfo files, netCDF files, …

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

We are connecting to a remote MySQL database (Specify).

It stores coordinates as numbers – non-spatial

We will use an OGR VRT file, XML defining

what sort of remote database? (Specify uses MySQL)
the connection parameters for the database (server, port, user, etc)
the SQL to run there to retrieve the required data
how to create a geometry from these data
(build a point feature from the original lon/lat numeric columns)

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

The XML VRT file...

```
<OGRVRTDataSource>
<OGRVRTLayer name="collectionObjects">
<SrcDataSource>MYSQL:niwainvert,user=*,password=*,host=*,port=*</SrcDataSource>
<SrcSQL>select …
…
...</SrcSQL>
<GeometryType>wkbPoint</GeometryType>
<GeometryField encoding="PointFromColumns" x="x" y="y"/>
<LayerSRS>EPSG:4326</LayerSRS>
</OGRVRTLayer>
</OGRVRTDataSource>
```

Data source, connection details, the SQL to run, how to build the geometry…

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

We can use the ogr_fdw extension to create a foreign table

This will use the VRT file as the data source, not as the data, but providing the information describing where to get the data when we query our foreign table.

The SQL to create the foreign table:

```
create extension ogr_fdw;

CREATE SERVER specify_vrt
FOREIGN DATA WRAPPER ogr_fdw
OPTIONS (datasource '/tmp/SpecifyQuery3.vrt', format 'OGR_VRT',
        updateable 'false' );
```

Foreign Data Wrappers

```
CREATE FOREIGN TABLE seamount_dev.fdw_collectionobjects
        (
         fid                    bigint,
         geom                   geometry(Point,4326),
         catalognumber          varchar(96),
         y                      double precision,
         x                      double precision,
         startdate              date,
         taxonname              varchar(765),
         ...
          )
        SERVER specify_vrt
        OPTIONS (layer 'collectionObjects');
```

This defines the columns in the table, matching those returned by the SQL in the VRT file

NIWA
Taihoro Nukurangi

Foreign Data Wrappers

As we did with the trawl foreign tables, we can now run an SQL to join local seamount data with the remote collection data:

```
select s.catalognum,
        s.taxonname,
        p.seamount,
        m.name
from fdw.specify s,
      seamount_polygons p,
      seamount m
where ST_Intersects(p.poly, ST_Transform(s.geom,3994))
   and p.seamount=m.reg_no;
```

catalognum	taxonname	seamount	name
000000005	Monachometra kermadecensis	907	Hinetapeka
000000039	Semitaspongia pulvinata	432	
000000158	Ircinia turrita	326	
000000352	Decapoda	1461	
000000446	Kemphyra corallina	138	Devonport Seamount
000000450	Nematocarcinus gracilis	751	Mt Ghost
000000472	Comatulides dawsoni	1478	SM6
000000473	Comatulides dawsoni	1478	SM6
000000597	Lepidopora dendrostylus	544	Tuatoru Knoll

NIWA
Taihoro Nukurangi

Summary

Using PostgreSQL foreign data wrappers, data stored in remote databases, files, even WFS services can be configured as foreign tables in a Postgres database.

Analyses that combine data from multiple locations, that require multiple extracts and merging can be replaced with a single SQL that returns the result in seconds.

NIWA
Taihoro Nukurangi

Thank you for attending!

Questions?