Database documentation: marlin
A metadatabase for Ministry for Primary Industries research data

P. M. Marriott
H. Sui

NIWA Fisheries Data Management
Database Documentation series

Revised Sep 2017

Version Control

Version	Changed By	Reason	Date
0.1	K. A. Mackay	Initial version, as NIWA Internal Report No. 90	Aug 2000
1.0	P. Marriott H. Sui	Updated database	Nov 2003
1.1	P.Marriott	Revised Datatypes section	May 2004
1.2	D Fisher	Added 'ctd' to Appendix 1 & section 2.4 on the web interface.	Mar 2006
1.3	D Fisher	Datatypes updated in Appendix 1	Oct 2009
1.4	D Fisher	Referenced database as metadb	July 2011
1.5	D Fisher	Updated MFish to MPI	Dec 2015
1.6	D Fisher	Updated dataset_species comment, Section 4	Sep 2017

Contents

List of Figures

1 Database documentation series

The National Institute of Water and Atmospheric Research (NIWA) currently carries out the role of Data Manager and Custodian for the fisheries research data owned by the Ministry for Primary Industries (MPI) formerly the Ministry of Fisheries.

This MPI data set, incorporates historic research data, data collected by MAF Fisheries prior to the split in 1995 of Policy to the Ministry of Fisheries and research to NIWA, and data collected by NIWA and other agencies for the Ministry of Fisheries and subsequently for MPI.

This document describes the Ministry of Fisheries research data metadatabase **marlin**, and is a part of the database documentation series produced by NIWA.

All documents in this series include an introduction to the database design, a description of the main data structures accompanied by an Entity Relationship Diagram (ERD), and a listing of all the main tables. The ERD graphically shows how all the tables link together.

This document is intended as a guide for users and administrators of the **marlin** database.

Access to this database and data is restricted to Nominated Personnel as specified in the current Data Management contract between the Ministry and NIWA. Any requests for data should in the first instance be directed to the Ministry.

2 Fisheries research metadata

2.1 Background

In the past few years, the use of metadata to describe data holdings by research organisations has gradually been accepted as a necessary tool for locating and describing datasets. There is a paucity of appropriate examples of the use of metadata in research organisations, especially relating to marine spatial datasets that NIWA specialises in, forcing NIWA to look elsewhere for relevant metadata standards. Within Australia, initiatives such as the Ocean Rescue 2000 "Blue Pages" theme directory for marine and coastal datasets (AODC, 1996) as well as the Australia New Zealand Land Information Council (ANZLIC)'s developing regional standard for geospatial metadata (ANZLIC, 1996-8) have stimulated organisations such as CSIRO Marine Research to start to collect metadata according to a local standard using metadata elements and terminology compliant with the ANZLIC metadata standard and the "Blue Pages".

In 1997 CSIRO Marine Research developed an in-house metadatabase termed the Marine Laboratories Information Network or "MarLIN" (Rees & Ryba, 1998). MarLIN was based on the pre-existing "Environmental Data Directory" (EDD or "Green Pages") software developed by Environment Australia. After a demonstration by CSIRO Marine Research in mid-2000, NIWA and the Ministry of Fisheries received permission to adopt MarLIN as the basis for their metadatabase for the fisheries research data. MarLIN, in a markedly modified form, is implemented as a relational database called **marlin** by NIWA, who act as the MPI fisheries data

custodians.

2.2 Nomenclature

- MarLIN refers to CSIRO's research data metadatabase.
- **marlin** is the highly modified physical implementation of MarLIN.
- **metadb** is the name of the database as implemented in Postgres, but since this database is commonly referred to as 'marlin' after the name of the internet web interface, this document refers to the database as marlin.

2.3 Database interface

General access is most readily available to authorised users through the Delphi application front end of the database. This front end is a user-friendly interface with the **marlin** database and does not require knowledge of SQL theory to interrogate the database. Communication between the application and the database is via ODBC (Open Database Connectivity)[1]. The Marlin application is written in Delphi and uses a two-tier architecture with database server and client tiers. All the business logic is implemented in the application.

2.4 Internet web interface

Access is available to this database on the internet via the url http://marlin.niwa.co.nz/ .
This web interface to marlin includes the following features:

1. A searchable catalogue of the data sets, originating from Ministry contracted scientific research projects that are now stored in Ministry data repositories. These records are searchable by a variety of fields including data type, species code and project code.

2. A searchable catalogue of aging material (otoliths), length frequency samples, and conversion factor tests, collected by fisheries observers. These records are searchable by species code, area and date (and processed state in the case of conversion factor tests).

3. A description of other types of scientific research data held by the Ministry that are not catalogued elsewhere within MARLIN.

4. A list of the codes used within New Zealand to describe species and areas.

[1] ODBC is an industry standard that allows communication between different database management systems.

3 Data structures

3.1 Database description

This database contains several tables. The ERD for **marlin** (Figure 1) shows the logical structure of the database and its entities (each entity is implemented as a database *table*) and relationships between these tables. Each table represents an object, event, or concept in the real world that is selected for representation in the database. Each *attribute* of a table is a defining property or quality of the table.

All of the table's attributes are shown in the ERD. The underlined attributes represent the table's primary key[2]. This schema is valid regardless of the database system chosen, and it can remain correct even if the Database Management System (DBMS) is changed. Most of the tables in this database have some attributes, called foreign keys[3], which contain standard NIWA fisheries codes, such as species. These attributes provide links to supporting tables within the **marlin** database.

Section 5 shows a listing of all the **marlin** tables as implemented by the Postgres DBMS. As can be seen in the listing of the tables, each table has a primary key. Primary keys are generally listed using the format:

Indexes: tablename_pkey primary key btree (attribute [, attribute])

Where:	tablename_pkey	is the name of the primary key index
	btree	is the index type
	attribute [, attribute]	are the attribute(s) the makes up the primary key (the key attributes).

A primary key prevents records with duplicate key values from being inserted into the table; e.g., a dataset with an existing dataset identifier.

For example, the primary key for the table *dataset* is shown as thus:

Indexes: dataset_pkey primary key btree (dataset_id)

The **marlin** database is implemented as a relational database. That is, each table is a special case of the mathematical construct known as a *relation* and hence elementary relation theory is used to deal with the data within tables and the relationships between them. There are both one-to-many and many-to-many relationships in **marlin.**

One-to-many relationships can be either mandatory or optional. These relationships are enforced

2 A primary key is an attribute or a combination of attributes that contains a unique value to identify that record.

3 A foreign key is any attribute, or a combination of attributes, in a table that is a primary key of another table. Tables are linked together through foreign keys.

in the database by the use of referential constraints[4]. Foreign key constraints do not allow *orphans* to exist in any table; i.e., where a child record exists without a related parent record. This may happen when:

i) a parent record is deleted;
ii) the parent record is altered so the relationship is lost;
ii) a child record is entered without a parent record.

All constraints in **marlin** prevent the latter from occurring. Constraints are shown in the table listings by the following format:

Foreign Key constraints: tablename_fk FOREIGN KEY (attribute) REFERENCES tablename (attribute)

Where:	tablename_fk	is the name of the foreign key constraint
	attribute	are the attribute the makes up the foreign key
	tablename (attribute)	is the primary key of the reference table

For example, a foreign key constraint on the attribute *data_type_id* in the table *dataset* is shown as thus:

Foreign Key constraints: dataset_types_fk FOREIGN KEY (data_type_id) REFERENCES data_types(data_type_id)

3.2 Database Design

As reflected by the ERD, the main table of **marlin** is the individual data set record table, *dataset* (Table 1). Each data set is uniquely identified by an integer, stored as the attribute *dataset_id*. What constitutes the definition of a dataset is deliberately left loosely defined, but generally, it is the aggregation of data based on some common element such as research voyage, project, habitat, species, etc. Attributes of the *dataset* table are open in their data types by having large character fields so as to allow more qualitative and descriptive data entry.

For all tables, each record is identified by one or two unique integers that make up the tables primary key(s).

Projects result in one or many data sets. Details for individual projects are stored in the *projects* (Table 2) table. It is assumed that projects may be interdisciplinary and involve more than one organisation. Multiple organisations can be associated with a project through the linking table *reference_projects*.

[4] Also known as integrity checks.

Details for individual people with access to, and/or management control over, data in the database are stored in the table *persons* (Table 3). General details for people are recorded here, such as database role, name, title, mailing address, street address, phone and fax numbers, and e-mail address.

The table *roles* (Table 4) defines the person's level of access to the marlin database. Currently this is set to 'administrator' with full editing access and 'user' with limited access privileges. The 'user' category may in the future be expanded to allow varying levels of access for different individuals. This may also include locking out various fields, tables or records.

The link between *persons* and *organisations* is through the table *positions* (Table 5). The position defines the person's role within the organisation. Conceptually, one person can fill positions within more than one organisation, and an organisation can have many people filling a position. The table positions records one relationship between a person and an organisation.

The table *data_types* (Table 7) contains a list of available types of data for an associated dataset. All datasets must be associated with only one data type. Examples are 'acoustic' which is acoustic data loaded to the 'acoustic' database on 'snapper', or 'otoliths' which are physical otolith specimens and will be stored in the NIWA otolith collection at Greta Point, Wellington.

Pre-defined geographic regions are stored in the *regions* table (Table 8). Multiple geographic regions can be associated with a dataset through the linking table *dataset_regions*. The original specifications for MarLIN, as designed by CSIRO Marine Research, called for generic geographic regions such as oceans and seas to be stored and defined by a minimum-bounding rectangles (MBRs) defined by north, south, east, and west coordinates. However, MBRs are unsuitable for defining many of the complex geographic regions in the *regions* table. So a future modification would be to link a dataset in the *dataset* table directly to an area polygon in the *map* table of the **rdb** database so that spatial queries can be performed directly. It is intended that the *regions* table be updated from the **rdb** database *area_codes* table regularly by automatic routine in a future upgrade.

Research voyages are stored in the *voyages* table (Table 9). Multiple voyages can be associated with a dataset through the linking table *dataset_voyages*.

The platform from which the data for the dataset was obtained is held in the *platforms* table (Table 10). Multiple platforms can be associated with a dataset through the linking table *dataset_platforms*. The term platform is used as data may have been collected from objects other than a vessel, this enables the table to contain information on other platforms of research such as fixed wharfs, airplanes (for aerial sighting of tuna schools, remotely operated vehicles (R.O.V's) etc. The platforms tables records details by storing the platform type (e.g. "Ship" or "Aircraft") and platform name (e.g., "FV Tangaroa" or "ZK-FDG").

marlin uses the MPI standard 3-character species codes. These codes are stored in the *species* table (Table 11). Multiple species can be associated with a dataset through the linking table *dataset_species*. This table is updated by a simple script run by the species codes data manager, currently the MPI Fisheries Data Manager, on an ad hoc basis, typically after new species records are added to the **rdb** database *species_master* table. It is intended that the *species* table be updated

from the **rdb** database *species_master* table regularly by automatic routine in a future upgrade.

The tables *dataset_platforms* (Table 16), and *dataset_species* (Table 18) are classic examples of how to resolve many-to-many relationships in a relational database. In each of these cases, one defined platform/species can be represented in many data sets, and one data set can have many defined platforms/species. To resolve this, these intermediate tables have been created to explicitly store each instance of a data set and defined platform/species relation. Other examples in this database are *dataset_regions*, *dataset_keywords*, *person_positions*, *reference_projects*, *dataset_projects* and *dataset_voyages*.

The final table is *t_data_desc* (Table 19). This table is unassociated with the rest of the database. It comprises descriptions of the various fields in the tables of the **marlin** database. It implements itself in the front end of the database, as a text box containing a description of a field as the cursor is held over it. Its function is to provide a ready reference of a fields intended content for users of the database.

3.3 Standards for fisheries databases

The **marlin** database was created in mid-2000 as more-or-less a direct copy of the MarLIN system developed by CSIRO Marine Research. In 1993, a set of standards was set in place (Ng 1992) for all fisheries databases. The most significant effect of these standards has been the requirement of adding of the prefix "t_" to the table names and "v_" to view names. However, this raised some potentially serious issues. The **marlin** database is just one part of a metadata system including CGI scripts and HTML pages. Modifying **marlin** to meet these standards would therefore have a very significant flow-on effect to all relevant fisheries applications. Therefore, table names in **marlin** have been intentionally left as written and do not meet fisheries standards.

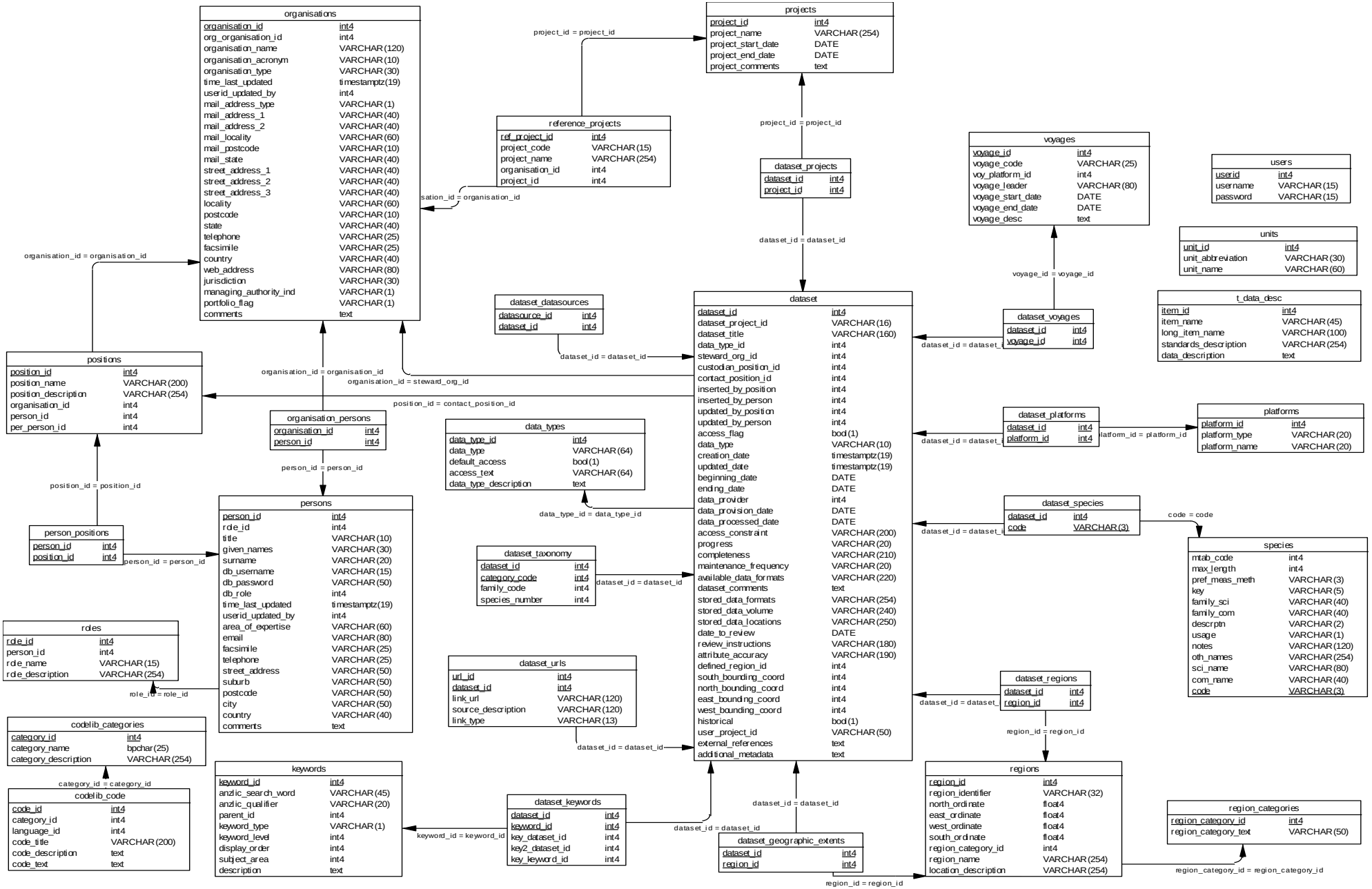

Figure 1: Entity Relationship Diagram (ERD) of the marlin database.

4 Table summaries

The **marlin** database has 19 tables currently utilised for containing metadata, the following is a listing and brief outline of the tables:

1. **dataset :** contains profile metadata information for individual datasets.
2. **projects :** contains details about the various projects associated with datasets.
3. **persons :** contains details about the various people associated with datasets.
4. **roles :** contains details about a persons level of access to **marlin**.
5. **positions :** contains details about positions that people fill within organisations.
6. **organisations :** contains details about the various organisations associated with datasets.
7. **data_types :** contains a list of available types of data for an associated dataset.
8. **regions :** contains brief descriptions of pre-defined geographical regions (e.g., oceans, statistical areas, QMA's, FMA's).
9. **voyages :** contains details of individual voyages resulting in datasets.
10. **platforms :** contains details of the platforms used to collect data for the datasets.
11. **species :** contains details of species associated with datasets.
12. **dataset_projects :** records the relationships between *dataset* and *projects*.
13. **reference_projects :** records the relationships between *projects* and *organisations*.
14. **person_positions :** records the relationships between *persons* and *positions*.
15. **dataset_voyages :** records the relationships between *dataset* and *voyages*.
16. **dataset_platforms :** records the relationships between *dataset* and *platforms*.
17. **dataset_regions :** records the relationships between *dataset* and *regions*.
18. **dataset_species :** records the relationships between *dataset* and *species*.
19. **t_data_desc :** provides a descriptive text of a field's intended content to be used in the client application.

A further 12 tables have been retained in the database schema pending possible future implementation. These are:

20. **region_categories :** contains a region categorization code (e.g. FMA, QMA, PAU7 subcode).
21. **dataset_urls :** contains URL links pertaining to individual datasets.
22. **keywords :** contains keywords (words or phrases summarizing aspects of the dataset), keyword types, and their subject areas.
23. **dataset_keywords :** records the relationships between *dataset* and *keywords*.
24. **codelib_categories** : this table is related to the keywords table as a reference to the code libraries used in the Keywords search function.
25. **codelib_code** : this table is related to the keywords table as a reference to the code libraries used in the Keywords search function.

26. **dataset_datasources** : this is a means of tracking the origins of datasets. Fields within the *dataset* table have now superseded the *dataset_datasources* table.
27. **dataset_geographic_extents :** this is a redundant table in the current database schema. It records the relationships between dataset and regions.
28. **dataset_taxonomy :** this table was utilised to link species data with dataset data. The *species* table has now superseded the *dataset_taxonomy* table.
29. **organisation_persons :** this is a redundant table in the current database schema. It records the relationships between the organisations, positions and persons tables.
30. **units :** holds standard descriptions and formats of units associated with a dataset.
31. **users :** a table that was used to control user access to the database.

5 marlin tables

The following are listings of the tables in the **marlin** database, including attribute names, data types (and any range restrictions), and modifiers.

Table 1: dataset

Column	Type	Description
dataset_id	integer	Not null. Unique identifier
dataset_project_id	character varying(16)	Not used
dataset_title	character varying(160)	Not null. Dataset full title
data_type_id	integer	Unique identifier refer 'data_types' table
steward_org_id	integer	Unique identifier refer 'organisations' table
custodian_position_id	integer	Unique identifier refer 'positions' table
contact_position_id	integer	Unique identifier refer 'positions' table
inserted_by_position	integer	Unique identifier refer 'positions' table
inserted_by_person	integer	Unique identifier for person inserting the dataset refer 'persons' table
updated_by_position	integer	Unique identifier for person updating the dataset refer 'positions' table
updated_by_person	integer	Unique identifier refer 'persons' table
access_flag	boolean	Tick flag showing if public access is allowed
data_type	character varying(10)	Unused
creation_date	timestamp with time zone	Date and time record was created
updated_date	timestamp with time zone	Date and time record was updated
beginning_date	date	Date when the dataset was implemented
ending_date	date	Date at which the dataset was closed
data_provider	integer	Unique identifier refer 'organisations' table
data_provision_date	date	Date when the dataset was supplied to the data services group
data_processed_date	date	Date when the dataset was processed by the data services group

Table 1: dataset (cont…)

Column	Type	Description
access_constraint	character varying(200)	Description of any access constraints
progress	character varying(20)	States whether a dataset is ‘in progress’ or ‘complete’
completeness	character varying(210)	States what is outstanding on the dataset
maintenance_frequency	character varying(20)	Not used
available_data_formats	character varying(220)	Description of the format that the dataset is stored in. EXCEL files, otoliths etc
dataset_comments	text	Open text field for comments and details
stored_data_formats	character varying(512)	Not used
stored_data_volume	character varying(240)	Dataset volume in MB, No.s of otoliths, No.s of station records etc
stored_data_locations	character varying(250)	Where the data the dataset pertains to is stored
date_to_review	date	Not used
review_instructions	character varying(180)	Not used
attribute_accuracy	character varying(190)	Not used
defined_region_id	integer	Not used
south_bounding_coord	integer	Not used
north_bounding_coord	integer	Not used
east_bounding_coord	integer	Not used
west_bounding_coord	integer	Not used
historical	boolean	States whether a dataset is derived from a current project or from a block of general submitted data
user_project_id	character varying(50)	Not used
external_references	text	References associated journal articles, reports etc
additional_metadata	text	Not used

Indexes:

dataset_pkey primary key btree (dataset_id)

Foreign Key constraints:

dataset_types_fk FOREIGN KEY (data_type_id) REFERENCES data_types(data_type_id)
dataset_organisations_fk FOREIGN KEY (steward_org_id) REFERENCES organisations(organisation_id)

Table 2: projects

Column	Type	Description

project_id	integer	Not null. Unique identifier
project_name	character varying(255)	Not null. The descriptive name of the project
project_start_date	date	The date project inception
project_end_date	date	The date of project closure
project_comments	text	Open text field

Indexes:

projects_pkey primary key btree (project_id)

Table 3: persons

Column	Type	Description
person_id	integer	Not null. Unique identifier
title	character varying(10)	Title e.g. Mr, Mrs, Ms etc..
given_names	character varying(30)	Not null. First name
surname	character varying(20)	Not null. Last name
db_username	character varying(15)	Database login name
db_password	character varying(50)	Not null. Login password
db_role	integer	Not null. Unique identifier as per table ‘role’
time_last_updated	timestamp with time zone	Date and time that the person record was updated
userid_updated_by	integer	Not null. Unique identifier as per person_id
area_of_expertise	character varying(60)	Descriptive comment
email	character varying(80)	Email address
facsimile	character varying(25)	Fax number
telephone	character varying(25)	Telephone number
street_address	character varying(50)	Contact address
suburb	character varying(50)	Contact address
postcode	character varying(50)	Contact address
city	character varying(50)	Contact address
country	character varying(40)	Contact address
comments	text	Free text field

Indexes:

persons_pkey primary key btree (person_id)

Foreign Key constraints:

persons_role_fk FOREIGN KEY (db_role) REFERENCES roles(role_id)

Table 4: roles

Column	Type	Description
role_id	integer	Not null. Unique identifier
role_name	character varying(15)	Not null. Persons database role
role_description	character varying(255)	Short description of persons database role

Indexes:
roles_pkey primary key btree (role_id)

Table 5: positions

Column	Type	Description
position_id	integer	Not null. Unique identifier
position_name	character varying(200)	Not null. Persons job position in their parent company
position_description	character varying(255)	Short description of their position
organisation_id	integer	Unique identifier refer table ‘organisations’
person_id	integer	Unique identifier refer table ‘persons’
per_person_id	integer	Unused

Indexes:
positions_pkey primary key btree (position_id)

Foreign Key constraints:
positions_organisations_fk FOREIGN KEY (organisation_id) REFERENCES organisations(organisation_id)

Table 6: organisations

Column	Type	Description
organisation_id	integer	Not null. Unique identifier
org_organisation_id	integer	Unused
organisation_name	character varying(120)	Not null. Company name
organisation_acronym	character varying(10)	Company acronym
organisation_type	character varying(30)	Organisation type
time_last_updated	timestamp with time zone	Date and time record was updated
userid_updated_by	integer	Who updated the record
mail_address_type	character varying(1)	Unused
mail_address_1	character varying(40)	Mailing address
mail_address_2	character varying(40)	Mailing address
mail_locality	character varying(60)	Mailing address
mail_postcode	character varying(10)	Mailing address
mail_state	character varying(40)	Unused
street_address_1	character varying(40)	Street address
street_address_2	character varying(40)	Street address
street_address_3	character varying(40)	Street address
locality	character varying(60)	Street address
postcode	character varying(10)	Street address
state	character varying(40)	Unused
telephone	character varying(25)	Telephone contact number
facsimile	character varying(25)	Fax contact number
country	character varying(40)	Country name
web_address	character varying(80)	Organisations www address
jurisdiction	character varying(30)	Unused
managing_authority_ind	character varying(1)	Unused
portfolio_flag	character varying(1)	Unused

Table 6: organisations (cont…)

Column	Type	Description
comments	text	Free text field

Indexes:
organisations_pkey primary key btree (organisation_id)

Table 7: data_types

Column	Type	Description
data_type_id	integer	Not null. Unique identifier
data_type	character varying(64)	Not null. Data type name
default_access	boolean	Unused
access_text	character varying(64)	Unused
data_type_description	text	Description of the data type

Indexes:
metadata_types_pkey primary key btree (data_type_id)

Table 8: regions

Column	Type	Description
region_id	integer	Not null. Unique identifier
region_identifier	character varying(32)	Not null. Region code. Refer NIWA's rdb: 'area_codes'
north_ordinate	real	Unused
east_ordinate	real	Unused
west_ordinate	real	Unused
south_ordinate	real	Unused
region_category_id	integer	Unused
region_name	character varying(255)	Not null. Region full name
location_description	character varying(255)	Description of region

Indexes:
regions_pkey primary key btree (region_id)

Foreign Key constraints:
regions_fk FOREIGN KEY (region_category_id) REFERENCES region_categories(region_category_id)

Table 9: voyages

Column	Type	Description
voyage_id	integer	Not null. Unique identifier
voyage_code	character varying(25)	Not null. As per NIWA standard voyage codes in the Central data files
voy_platform_id	integer	Unique identifier refer table 'platforms'
voyage_leader	character varying(80)	Name of voyage leader
voyage_start_date	date	Date of voyage departure
voyage_end_date	date	Date of voyage return
voyage_desc	text	Open text field

Indexes:
voyages_pkey primary key btree (voyage_id)

Table 10: platforms

Column	Type	Description
platform_id	integer	Not null. Unique identifier
platform_type	character varying(20)	Platform type e.g. boat, wharf, airplane etc..
platform_name	character varying(20)	Not null. Name of the platform

Indexes:
platforms_pkey primary key btree (platform_id)

Table 11: species

Column	Type	Description
mtab_code	integer	Unused
max_length	integer	Normal maximum length obtained by the species
pref_meas_meth	character varying(3)	Preferred measurement method for measuring this species length
key	character varying(5)	Identification key. First character represents the type of fish: B=Bony fish; C= Cartilaginous fish (sharks and rays). Numbers refer to the genera identification key in Paulin C, Stewart A, Roberts C, McMillan P. 1989. New Zealand Fish. A complete Guide. 279p. ISBN 0-477-01427-5
family_sci	character varying(40)	Scientific family name
family_com	character varying(40)	Common family name
descrptn	character varying(2)	Brief description of species - fish, shellfish, etc.
usage	character varying(1)	Describes whether code is for ITQ, Research etc. Usage O=Obsolete code (refer rdb: species_usage).
notes	character varying(120)	Any notes about features, peculiarities etc of the species
oth_names	character varying(255)	Other names associated with the species
sci_name	character varying(80)	Scientific name for the species
com_name	character varying(40)	Common name for the species
code	character varying(3)	Not null. Unique identifier. Three character (upper case) code for the species

Indexes:

species_pkey primary key btree (code)

Table 12: dataset_projects

Column	Type	Description
dataset_id	integer	Not null. Unique identifier refer 'dataset' table
project_id	integer	Not null. Unique identifier refer 'projects' table

Indexes:

dataset_projects_pkey primary key btree (dataset_id, project_id)

Foreign Key constraints:

dataset_project_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_project_pfk FOREIGN KEY (project_id) REFERENCES projects(project_id)

Table 13: reference_projects

Column	Type	Description
ref_project_id	integer	Not null. Unique identifier
project_code	character varying(15)	Not null. Organisations alphanumeric code for the project
project_name	character varying(255)	Organisations name for the project
organisation_id	integer	Unique identifier refer ‘organisations’ table
project_id	integer	Unique identifier refer ‘projects’ table

Indexes:

reference_projects_pkey primary key btree (ref_project_id)

Foreign Key constraints:

reference_pro_pfk FOREIGN KEY (project_id) REFERENCES projects(project_id)
reference_pro_ofk FOREIGN KEY (organisation_id) REFERENCES organisations(organisation_id)

Table 14: person_positions

Column	Type	Description
person_id	integer	Not null. Unique identifier refer ‘persons’ table
position_id	integer	Not null. Unique identifier refer ‘positions’ table

Indexes:

person_positions_pkey primary key btree (person_id, position_id)

Foreign Key constraints:

person_positions_pfk FOREIGN KEY (person_id) REFERENCES persons(person_id)
person_positions_qfk FOREIGN KEY (position_id) REFERENCES positions(position_id)

Table 15: dataset_voyages

Column	Type	Description
dataset_id	integer	Not null. Unique identifier refer ‘dataset’ table
voyage_id	integer	Not null. Unique identifier refer ‘voyages’ table

Indexes:

dataset_voyages_pkey primary key btree (dataset_id, voyage_id)

Foreign Key constraints:

dataset_voyages_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_voyages_vfk FOREIGN KEY (voyage_id) REFERENCES voyages(voyage_id)

Table 16: dataset_platforms

Column	Type	Description
dataset_id	integer	Not null. Unique identifier refer ‘dataset’ table
platform_id	integer	Not null. Unique identifier refer ‘platforms’ table

Indexes:

dataset_platforms_pk primary key btree (dataset_id, platform_id)

Foreign Key constraints:

dataset_platforms_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_platforms_vfk FOREIGN KEY (platform_id) REFERENCES platforms(platform_id)

Table 17: dataset_regions

Column	Type	Description
dataset_id	integer	Not null. Unique identifier refer ‘dataset’ table
region_id	integer	Not null. Unique identifier refer ‘regions’ table

Indexes:

dataset_regions_pkey primary key btree (dataset_id, region_id)

Foreign Key constraints:

dataset_regions_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_regions_rfk FOREIGN KEY (region_id) REFERENCES regions(region_id)

Table 18: dataset_species

Column	Type	Description
dataset_id	integer	Not null. Unique identifier refer ‘dataset’ table
code	character varying(3)	Not null. Unique identifier refer ‘species’ table

Indexes:

dataset_species_pkey primary key btree (dataset_id, code)

Foreign Key constraints:

dataset_species_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_species_sfk FOREIGN KEY (code) REFERENCES species(code)

Table 19: t_data_desc

Column	Type	Description
item_id	integer	Not null. Unique identifier
item_name	character varying(45)	Field name in the marlin application
long_item_name	character varying(100)	Unused
standards_description	character varying(256)	Unused
data_description	text	Text describing the function and/or range of data a field can contain

Indexes:

t_data_desc_pkey primary key btree (item_id)

The following tables are not currently in use, but are retained in the database schema pending possible future implementation.

Table 20: region_categories

Column	Type	Description
region_category_id	integer	Not null. Unused
region_category_text	character varying(50)	Unused

Indexes:
region_categories_pkey primary key btree (region_category_id)

Table 21: dataset_urls

Column	Type	Description
url_id	integer	Not null. Unused
dataset_id	integer	Not null. Unused
link_url	character varying(120)	Unused
source_description	character varying(120)	Unused
link_type	character varying(13)	Unused

Indexes:
dataset_urls_pkey primary key btree (dataset_id, url_id)

Foreign Key constraints:
dataset_urls_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)

Table 22: keywords

Column	Type	Description
keyword_id	integer	Not null. Unused
anzlic_search_word	character varying(45)	Unused
anzlic_qualifier	character varying(20)	Unused
parent_id	integer	Unused
keyword_type	character varying(1)	Unused
keyword_level	integer	Unused
display_order	integer	Unused
subject_area	integer	Unused
description	text	Unused

Indexes:

keywords_pkey primary key btree (keyword_id)

Table 23: dataset_keywords

Column	Type	Description
dataset_id	integer	Not null. Unused
keyword_id	integer	Not null. Unused
key_dataset_id	integer	Unused
key2_dataset_id	integer	Unused
key_keyword_id	integer	Unused

Indexes:

dataset_keywords_pkey primary key btree (dataset_id, keyword_id)

Foreign Key constraints:

dataset_keywords_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_keywords_vfk FOREIGN KEY (keyword_id) REFERENCES keywords(keyword_id)

Table 24: codelib_categories

Column	Type	Description
category_id	integer	Not null. unused
category_name	character(25)	Unused
category_description	character varying(255)	Unused

Indexes:
codelib_categories_pkey primary key btree (category_id),
codelib_categ_category_name_key unique btree (category_name)

Table 25: codelib_code

Column	Type	Description
code_id	integer	Not null. Unused
category_id	integer	Unused
language_id	integer	Default 1. Unused
code_title	character varying(200)	Unused
code_description	text	Unused
code_text	text	Not null. Unused

Indexes:
codelib_code_pkey primary key btree (code_id),
codelib_code_code_title_key unique btree (code_title)

Foreign Key constraints:
codelib_code_fk FOREIGN KEY (category_id) REFERENCES codelib_categories(category_id)

Table 26: dataset_datasources

Column	Type	Description
datasource_id	integer	Not null. Unused
dataset_id	integer	Not null. Unused

Indexes:
dataset_datasources_pkey primary key btree (datasource_id, dataset_id)

Foreign Key constraints:
dataset_datasources_fk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)

Table 27: dataset_geographic_extents

Column	Type	Description
dataset_id	integer	Not null. Unused
region_id	integer	Not null. Unused

Indexes:
dataset_geographic_extents_pkey primary key btree (dataset_id, region_id)

Foreign Key constraints:
dataset_geo_dfk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)
dataset_geo_rfk FOREIGN KEY (region_id) REFERENCES regions(region_id)

Table 28: dataset_taxonomy

Column	Type	Description
dataset_id	integer	Not null. Unused
category_code	integer	Not null. Unused
family_code	integer	Unused
species_number	integer	Unused

Indexes:
dataset_taxonomy_pkey primary key btree (dataset_id, category_code)

Foreign Key constraints:
dataset_taxonomy_fk FOREIGN KEY (dataset_id) REFERENCES dataset(dataset_id)

Table 29: organisation_persons

Column	Type	Description
organisation_id	integer	Not null. Unused
person_id	integer	Not null. Unused

Indexes:
organisation_persons_pkey primary key btree (organisation_id, person_id)

Foreign Key constraints:
organisation_persons_ofk FOREIGN KEY (organisation_id) REFERENCES organisations(organisation_id)
organisation_persons_pfk FOREIGN KEY (person_id) REFERENCES persons(person_id)

Table 30: units

Column	Type	Description
unit_id	integer	Not null. Unused
unit_abbreviation	character varying(30)	Unused
unit_name	character varying(60)	Unused

Indexes:
units_pkey primary key btree (unit_id)

Table 31: users

Column	Type	Description
userid	integer	Not null. Unused
username	character varying(15)	Unused
password	character varying(15)	Unused

Indexes:
users_pkey primary key btree (userid)

6 marlin business rules

6.1 Introduction to business rules

The following are a list of business rules pertaining to the **marlin** database. A business rule is a written statement specifying what the information system (i.e., any system that is designed to handle marlin survey data) must/should do, or how it must be structured.

Validation rules may be part of the preloading checks on the data as opposed to constraints or checks imposed by the database. These rules sometimes state that a value should be within a certain range. All such rules containing the word 'should' are conducted by the application front-end software. The use of the word 'should' in relation to these validation checks means that; i) the contents of the field are system generated by the application or ii) a warning message is generated when a value falls outside of the designated range for the field, and the update or implementation of a new record cannot be completed until the fields contents fall within acceptable limits.

There are three recognized types of business rules:

Fact	Certainty or an existence in the information system
Formula	Calculation employed in the information system
Validation	Constraint on a value in the information system

Referential constraints and range checks, both in the database and the application front-end, implement the formula and validation type rules.

6.2 Business rules

Table 1: dataset

dataset_id	Data set identifier must be unique integer greater than zero.
dataset_title	Must have a value entered.
data_type_id	Should be a valid data_type identifier as listed in the *data_types* table.
steward_org_id	Should be a valid organisation identifier as listed in the *organisations* table.
custodian_position_id	Should be a valid position identifier as listed in the *positions* table.
contact_position_id	Should be a valid position identifier as listed in the *positions* table.
inserted_by_position	Should be a valid position identifier as listed in the *positions* table.
inserted_by_person	Should be a valid person identifier as listed in the *persons* table.
updated_by_position	Should be a valid position identifier as listed in the *positions* table.
updated_by_person	Should be a valid person identifier as listed in the *persons* table.
access_flag	Should have a value ‘t’ or null.
creation_date	Should have a valid date.
updated_date	Should have a valid date.
beginning_date	Should be a valid date.
ending_date	Should be a valid date or null. **Multiple column check on beginning and end dates:** The data set ending date must be after the beginning date.
data_provider	Should be a valid organisation identifier as listed in the *organisations* table.
data_provision_date	Should be a valid date or null.
data_processed_date	Should be a valid date.

access_constraint	Should have a value entered.
progress	Should have a value either ‘in progress’ or ‘complete’
a**vailable_data_formats**	Should have a value
stored data_locations	Should have a value

Table 2: projects

project_id	Project identifier must be unique integer greater than zero.
project_name	Must have a value entered.
project_start_date	Should be a valid date format
project_end_date	Should be a valid date format or null
	Multiple column check on project start and end date: The project end date must be after the project start date.

Table 3: persons

person_id	Person identifier must be unique integer greater than zero.
given_names	Must have a value entered. First name
surname	Must have a value entered. Surname

Table 4: roles

role_id	Role identifier must be unique integer greater than zero.
role_name	Must have a value entered.

Table 5: positions

position_id	Position identifier must be unique integer greater than zero.
position_name	Must have a value entered.

Table 6: organisations

organisation_id	Organisation identifier must be unique integer greater than zero.
organisation_name	Must have a value entered.

Table 7: data_types

datatype_id	Data type identifier must be unique integer greater than zero.
data_type	Must have a valid value as listed in Appendix 1.

Table 8: regions

region_id	Region identifier must be unique integer greater than zero.
region_identifier	Must have a valid value compatible with the codes of NIWA's **rdb***: area_codes.*
region_name	Must have a value entered.

Table 9: voyages

voyage_id	Voyage identifier must be unique integer greater than zero.
voyage_code	Must have a valid value compatible with the voyage codes of NIWA's Central Data Files.
voyage_platform_id	Should be a valid platform identifier as listed in the *platforms* table.

Table 10: platforms

platform_id	Platform identifier must be unique integer greater than zero.
platfrom_type	Should have a value entered.
platform_name	Must have a value entered.

Table 11: species

family_sci	Should be equal to the value of the same field in the **rdb***:species_master* table for the same species code
family_com	Should be equal to the value of the same field in the **rdb***:species_master* table for the same species code
oth_names	Should be equal to the value of the same field in the **rdb***:species_master* table for the same species code
sci_name	Should be equal to the value of the same field in the **rdb***:species_master* table for the same species code
com_name	Should be equal to the value of the same field in the **rdb***:species_master* table for the same species code
code	Must be a valid code as listed in the **rdb***:species_master* table.

Table 12: dataset_projects

dataset_id	Must be a dataset identifier as listed in the *dataset* table.
project_id	Must be a valid project identifier as listed in the *projects* table.

Table 13: reference_projects

ref_project_id	Ref_project identifier must be unique integer greater than zero.
project_code	Must have a value, it should be the relevant organisations appropriate project code for the project name.
project_name	Should have a value entered.
organisation_id	Should be a valid organisation identifier as listed in the *organisations* table.
project_id	Should be a valid project identifier as listed in the *projects* table.

Table 14: person_positions

person_id	Must be a valid person identifier as listed in the *persons* table.
position_id	Must be a valid position identifier as listed in the *positions* table.

Table 15: dataset_voyages

dataset_id	Must be a valid dataset identifier as listed in the *dataset* table.
voyage_id	Must be a valid voyage identifier as listed in the *voyages* table.

Table 16: dataset_platforms

dataset_id	Must be a valid dataset identifier as listed in the *dataset* table.

platform_id	Must be a valid platform identifier as listed in the *platforms* table.

Table 17: dataset_regions

dataset_id	Must be a valid dataset identifier as listed in the *dataset* table.
region_id	Must be a valid region identifier as listed in the *regions* table.

Table 18: dataset_species

dataset_id	Must be a valid dataset identifier as listed in the *dataset* table.
code	Must be a valid code identifier as listed in the *species* table.

7 Acknowledgements

The authors would like to thank Kimon George and David Fisher and Fred Wei for their editorial contributions to this document, and Kevin Mackay for the original **marlin** documentation and comments on this edition.

8 References

Australia New Zealand Land Information Council (ANZLIC) (1996-8). Core Metadata Elements for Land and Geographic Directories in Australia and New Zealand. Digital document, available online at: http://www.anzlic.org.au/infrastructure_metadata.html

Australian Oceanographic Data Centre (AODC) (1996). The marine and coastal data directory of Australia – the Blue Pages, Version 1.0. available online at: http://www.marine.csiro.au/marine/mcdd/data/CSIRODMR/CSIRODMR_datasets.html

CSIRO Marine Research Datacentre. Available online at: http://www.marine.csiro.au/datacentre/

Ng, S. 1992: Standards for setting up databases and their applications. *MAF Fisheries Greta Point Internal Report No 180*. 31p.

Rees, A.J.J. and Ryba, M.M. (1998). MarLIN – a metadatabase for research data holding at CSIRO Marine Research. Paper presented at the first Australian Marine and Coastal Data Management Conference, Hobart, November 1998. Available online at: http://www.marine.csiro.au/datacentre/ext_docs/marlinpaper.htm

Appendix 1 – Metadata types Table

Short description	Long description	Restrictions
'acoustic' database	Electronic data from acoustic surveys.	Must have been stored in the 'acoustic' database.
aer_sight' database	Electronic data from aerial sightings of pelagic schooling species.	Must have been stored in the 'aer_sight' database.
'age' database	Electronic data from the aging of fish (for example - the reading of otoliths).	Must have been stored in the 'age' database.
'beach' database	Electronic data from surveys of beaches. For example - transect surveys.	Must have been stored in the 'beach' database.
Bibliography, electronic	??? Electronic data storing the results of literature searches ??? DESCRIPTION NEEDS CLARIFICATION	
'biods' database	Electronic data from bio-diversity or bio-security surveys. For example - Port and harbour surveys for invasive pest species	Must have been stored in the 'biods' database.
'ctd' database	Electronic data relating to oceanography including that collected using a CTD (Conductivity Temperature and Depth) probe.	Must have been stored in the 'ctd' database.
'dive' database	Electronic data from dive surveys. For example - random dive surveys, or fish aggregation surveys, or transect dive surveys.	Must have been stored in the 'dive' database.
'fish_ce database	Electronic data resulting from the grooming, by researchers, of commercial Catch and Effort data.	Must have been stored in the 'fish_ce' database. Must not be for the method purse seining. Must not be for surface longlining for tuna.
Fishing equipment data	Data (either electronic or physical) on the effectiveness or performance of equipment used in fishing. For example - data on hook sink rates.	
General Analysed Data Sets	Electronic data showing a researchers workings or analysis.	
General bio-div data	Data on the distribution of marine organisms of a form not suitable for loading onto the biods database. Stored on CD	Must be in a format incompatible with the 'biods' database. Must be stored on a CD.
General bio-sec data	Data on the distribution of invasive marine organisms of a form not suitable for loading onto the biods database. Stored on CD	Must be in a format incompatible with the 'biods' database. Must be stored on a CD.
General Recreational Data	Electronic data from recreational fishing surveys and questionnaires, of a format that could not be loaded into the 'rec_data' database.	Must be in a format incompatible with the rec_data database. Must be stored on a CD.
'genetics' databases	Electronic data (ASCII files) describing the genetic properties of tissue samples and used as an input to computer programmes which carry out genetic analysis.	Must have been stored in the 'genetics' database.
Groomed CatchEff Data	Electronic data resulting from the grooming (by researchers) of commercial catch and effort data - where such data was not in an appropriate format to load into the fish_ce, pseine, or tuna databases. Stored on CD.	Must be in a format incompatible with the fish_ce, pseine or tuna database. Must be stored on a CD.
Groomed Observer LFreq Data	Electronic data resulting from the grooming (by researchers) of observer length frequency data. Stored on CD.	Must be stored on a CD.
Groomed Observer LongLine data	Electronic data resulting from the grooming (by researchers) of observer data describing the fishing activity of tuna longline vessels. Stored on CD.	Must be in a format incompatible with the l_line database. Must be stored on a CD.

Groomed Observer Trawl data	Electronic data resulting from the grooming (by researchers) of observer data describing the fishing activity of trawlers. Stored on CD.	Must be in a format incompatible with the obs database. Must be stored on a CD.
'iki' database	Electronic data from biological sampling aboard commercial fishing vessels (commonly longliners but sometimes trawl or danish seiners).	Must have been stored in the 'iki' database.
Image/audio archive	Data (either electronic or physical) being primarily images or sound. For example - photographs, diagrams, illustrations, video and audio tapes.	
'kina' database	Electronic data from kina (Evechinus chloroticus) sampling surveys. Includes data from commercial fishers, catch samplers working aboard commercial fishing vessels and research dive surveys.	Must have been stored in the 'kina' database.
'l_line' database	Electronic data, collected by observers, describing the fishing activities of tuna longline vessels.	Must have been stored in the 'l_line' database.
'market' database	Electronic data from biological sampling in fish markets, processing sheds and on wharves.	Must have been stored in the 'market' database.
Merged datasets	Electronic datasets created as a result of the matching of data from two (or more) different types of electronic data sources. For example - merging catch effort data with observer data or survey data.	
'non_fish_bycatch'	Electronic data collected from commercial fishers describing the accidental capture of marine mammals and seabirds.	Must have been stored in the 'non_fish_bycatch' database.
'obs' database	Electronic data, collected by observers, describing the fishing activities of trawlers.	Must have been stored in the 'obs' database.
'obs_lfs' database	Electronic data, collected by observers aboard commercial fishing vessels, describing the biological characteristics (including length frequency, weight, gonad stage etc.) of fish.	Must have been stored in the 'obs_lfs' database.
Oceanography/habitat data	Data (physical or electronic) on the properties of the marine habitat, specifically ocean depth and the physical properties of the ocean floor.	
Other	Miscellaneous data that does not fit any other specified data type. This data type should only be used after consultation and agreement with the Research data management group at MPI.	Only to be used following the receipt of approval from the MPI research data management group.
'oyster' database	Electronic data from Foveaux Strait oyster (Tiostrea chilensis) sampling programmes.	Must have been stored in the 'oyster' database.
'plankton' database	Electronic data from plankton and egg surveys.	Must have been stored in the 'plankton' database.
'pseine' database	Electronic data resulting from the grooming, by researchers, of commercial purse seining Catch and Effort data. This primarily holds data on Kahawai and Jack Mackerel.	Must have been stored in the 'pseine' database.
Raw data forms	Physical data in the form of separate paper forms or bound books of paper forms, commonly being information collected during surveys or sampling programmes.	Must be paper, either loose leaf or bound. Must be a printed document with blank spaces for information to be inserted.
Raw electronic data	The raw electronic data collected during surveys or sampling programmes.	Must be the result of a survey or sampling programme. Must not be groomed. Must not be working files or analyses. Must be stored on CD.
'rec_data' database	Electronic data from recreational fishing surveys and questionnaires.	Must have been stored in the 'rec_data' database.
'recruit' database	Electronic data from kahawai (Arripis trutta) recruitment surveys.	Must have been stored in the 'recruit' database.

'rlcs' database	Electronic data from rock lobster (Jasus edwardsii and Jasus verreauxi) catch sampling programmes.	Must have been stored in the 'rlcs' database.
'rocklob' database	Electronic data from rock lobster (Jasus edwardsii and Jasus verreauxi) puerulus and phyllosoma sampling programmes.	Must have been stored in the 'rocklob' database.
'scallop' database	Electronic data from scallop dive and dredge surveys.	Must have been stored in the 'scallop' database.
Specimens, ageing	Parts of marine organisms collected primarily to assist in assessments of the age of that species. For example - otoliths. Otoliths are routinely inventoried in the 'age' database.	Must not be a whole organism. Must be collected primarily to aid in aging studies.
Specimens, non-ageing	Parts of marine organisms collected primarily for a purpose other than assessments of the age of that species. For example - tissue samples.	Must not be a whole organism. Must not be collected primarily to aid in aging studies.
Specimens, whole	Whole marine organisms.	Must be a whole organism.
'tag' database	Electronic data from tagging programmes.	Must have been stored in the 'tag' database.
'trawl' database	Electronic data from trawl surveys (including effort, catch and biological sampling).	Must have been stored in the 'trawl' database.
'tuna' database	Electronic data resulting from the grooming, by researchers, of commercial surface longlining (for tuna) Catch and Effort data.	Must have been stored in the 'tuna' database.
seamount' database	Electronic data presenting a synopsis of the physical characteristics of seamounts.	Must have been stored in the 'seamount' database.
GIS data	shape files	