Database documentation: tuna

F. Wei

NIWA Fisheries Data Management
Database Documentation Series

Revised Jan 2016

Revision History

Version	Change	Date	Responsible
1.0	First release	14 Jun 2002	Fred Wei
1.1	Added project_id in t_trip table, fma in t_effort table, moon_phase, quarter, bottom_depth and soi columns in t_effort and t_tlset tables.	30 Oct 2002	Fred Wei
1.1.1	Rename column name "key" to *tablename*_key.	20 Mar 2003	Fred Wei
1.2	added section 6.6 and 6.7, and BR20	10 Nov 2003	Fred Wei
2.0	Update for new TLCER old and new form	Mar 2004	Fred Wei
2.1	Add more comments in section 6.3	Dec 2004	Fred Wei
3.0	Big update according to new rules and re-groomed whole database	Mar-May 2005	Fred Wei
3.1	Update document to reflect that BR4 was deleted, also according to Kim George's email on 7/12/05 BR7 was removed. Added a bulletin under rule DS3 to remove un-relevant trips.	Jan 2007	Fred Wei
3.2	Postgres version	Jan 2016	D Fisher, F Wei

Contents

List of Figures/Tables

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 provides an introduction to the database **tuna**, and is a part of the database documentation series produced by NIWA. It supersedes previous documentation by H.A. Dean (1998) on this database.

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 and business rules. The ERD graphically shows how all the tables fit in together, and their relationships to other databases.

This document is intended as a guide for users and administrators of the **tuna** database.
This database has been implemented as a schema within the Postgres database called **fish**.

2 Commercial tuna data

2.1 Data sources

2.1.1 New Zealand Data

The data in **tuna** are derived from Ministry for Primary Industries catch and effort data and cover a period from 1 January 1980 to the present. Data compiled from before 1988 are sourced from the Fisheries Statistics Unit (FSU), a precursor to the Ministry of Fisheries catch and effort database. Recent data are derived from the Tuna Longlining Catch and Effort Return (TLCER, old and new form) and Catch and Effort Landing Return (CELR) forms.

Some scientific research data are also collected by commercial operators for the Real Time Monitoring Programme (RTMP), using the RTMP form, which collects single fish length and weight records.

Nearly all the commercially collected data can be linked to the Scientific Observer longline tuna database, **l_line,** which holds hook, catch, and environment data collected by New Zealand observers.

2.1.2 Australian Data

A series of independent tables hold data collected by Australian based observers for the Real Time Monitoring Programme (RTMP). (*These data have more detail than New Zealand commercial data and should ideally be held in the **l_line** related tables in **cod**.*)

2.1.3 Japanese Data

Data summaries provided by the Japanese are held in accordance with exchange protocols established by the Commission for the Conservation of the Southern Bluefin Tuna (CCSBT). These data are in the standard Yellow Book summary format, which summaries catch and effort by five-degree squares of

latitude and longitude and monthly periods of time.

3 Data structures

3.1 Introduction

The **tuna** database began as a collection of surface longlining (SLL) catch and effort logbooks from foreign licensed tuna fishing vessels in the 1980's. Science staff designed the logbooks and the data were stored in computer flat files accessed by a customized suite of Pascal programmes. When MFish Information & Technology Division (ITD) began, official records were archived and **tuna** became the only active source of historical New Zealand SLL tuna catch and effort records which are retained for scientific purposes. Scientific stock management requires the longest available time series to give more weight to Virtual Population Analyses (VPA), reducing probability bandwidths. When ITD began, a new series of data collection forms was instituted. These are now the current TLCER and CELR formats. TLCER data continued to be added to the **tuna** flat files; however some catch and effort data recorded by domestic fishers choosing to use CELR forms were omitted. In 1993 the current design and structure of **tuna** was implemented in an Empress Relational Database Management System (RDBMS). Flat file data were decoded and imported to the more efficient relational structures described in this document. This process has enabled the ability to marry data from TLCER and CELR sources. As other aspects of New Zealand's trilateral commitments to the CCSBT have increased, so has the capacity of **tuna** to include data from other forms including the supplementary RTMP data stored in the table *t_supp*, and the Japanese Yellow Book form in table *t_yellow*.

The data model for **tuna** is an accurate reflection of the real world of commercial tuna fishing within the New Zealand Exclusive Economic Zone (EEZ). Vessels make trips (*t_trip*) within the EEZ under licensing agreements for commercial tuna fishing. While fishing, all vessels are required by law to declare their catch and effort on TLCER or CELR forms. The TLCER form records surface longlining (SLL) effort (*t_tlset*), total catch (*t_tlcatch*) and individual Southern Bluefin Tuna (SBT) processing weights (*t_indcatch*). A table in parallel to *t_tlset*, *t_effort,* holds CELR effort data. Table *t_catch* holds estimated catch data for all methods. Tuna fishing methods mainly comprise SLL, Handlining (HL), Trolling (T), Trot lines or Dahn lines (TL), and Pole & Line (PL). An extra table *t_landing* holds processing and catch landing records from the MFish Catch Landing Record form (CLR) in the bottom half of the CELR form. This table is used to certify the catch reporting methods of the CELR form where fishers exercise the right to report number or weight of fish caught.

3.2 Database description

The ERD for **tuna** (Figure 1) shows the logical structure of the database and its entities (each entity is implemented as a database *table*) and relationships between these. All of the table's attributes are shown in the ERD. The underlined attributes represent the table's primary key[1]. This schema is valid regardless of the database system chosen, and it can remain correct even if the Database Management System (DBMS) is changed.

Each table represents an object, event, or concept in the real world that has been selected to be represented in the database. Each *attribute* of a table is a defining property or quality of the table.

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

Most of the tables in the **tuna** database also contain special attributes, called foreign keys[2].

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

Indices: index_name PRIMARY KEY, btree (*attribute* [, *attributes*])

This prevents records with duplicate key values from being inserted into the table, e.g., a trip with an existing trip key.

The **tuna** 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. All relationships in **tuna** are of the type *one-to-many*[3]. This is shown in the ERD by connecting a single line from the parent table; e.g., *t_trip*, to the child table; e.g., *t_effort*, with crows foot (indicating 'many) pointing to the child. For example, consider the relationship between the tables' *t_trip* (the parent table) and *t_effort* (the child table). Any one trip in *t_trip* can have one or more units of effort in *t_effort*, but any one unit of effort can only be a part of one trip. Note that the word 'many' applies to the possible number of records another is associated with. For a given instance, there might be zero, one, two, or more associated records, but if it is ever possible to have more than one, we use the word 'many' to describe the association.

Note that the one-to-many relationships can be either mandatory or optional. The optional relationship, denoted in the ERD by the symbol "O" at one or both ends of the relationship line, means that a record does not have to have any associated records. Conversely, the mandatory relationship denoted in the ERD by a bar symbol across the relationship line, means that a record has to have at least one associated record. For example, if we consider again the one-to-many relationship between the tables *t_trip* and *t_effort*, which has a mandatory 'one' and an optional 'many'. This means that one trip record can have zero or more (many) units of effort within it, but one unit of effort must have one, and only one, associated record in the trip table.

These relationships are enforced in the database by the use of foreign key constraints[4]. These 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;
iii. or a child record is entered without a parent record.

[2] 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.
[3] A one-to-many relationship is where one record (the *parent*) in a table relates to one or many records (the *child*) in another table; e.g., one survey in *t_survey* can have many strata in *t_stratum* but any one stratum can only come from one survey.
[4] Also known as referential constraints or integrity checks.

All constraints in **tuna** prevent these from occurring. Foreign key constraints are shown in the table listings by the following example:

Foreign-key constraints:
"fk_t_effort_t_trip" FOREIGN KEY (trip_key) REFERENCES t_trip(trip_key)

This means that the value of the attribute *trip_key* in the current record must already exist in the parent table *t_trip* or the record will be rejected and an error message will be displayed:

All tables in the database are indexed. That is, attributes that are most likely to be used as a searching key have like values linked together to speed up searches. These indices are listed using the following example:

vessel_key	BTREE	Normal

Details for each trip are held in the table *t_trip*. Each trip is uniquely identified by a trip code, stored as the attribute *trip_key*.

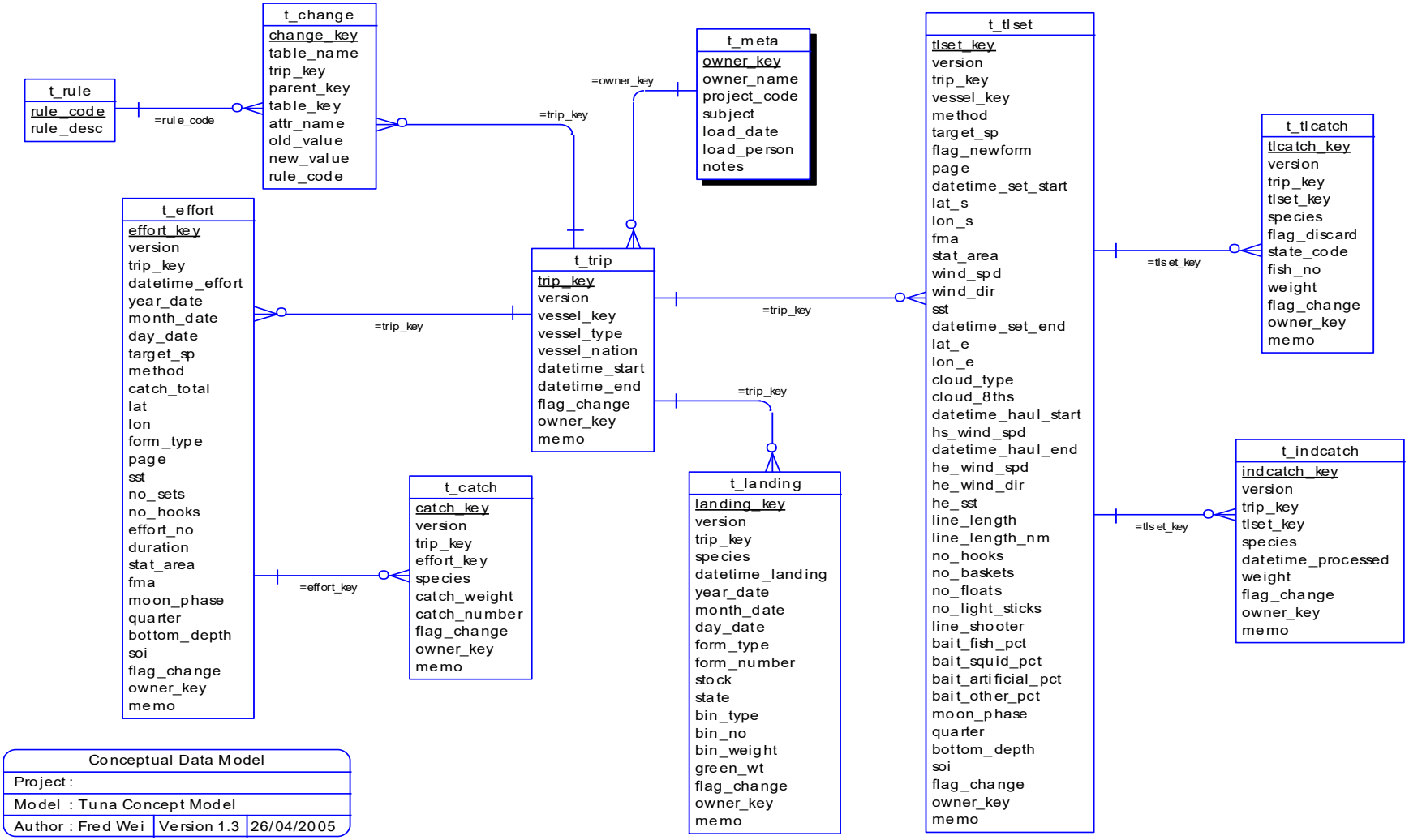

Figure 1: Entity Relationship Diagram (ERD) of the tuna database.

Supplementary data collected from fishers in the RTMP programme, are stored in *t_supp* (Figure 2). *t_supp* holds the length and weight of individual processed SBT and Bigeye tuna (BIG).

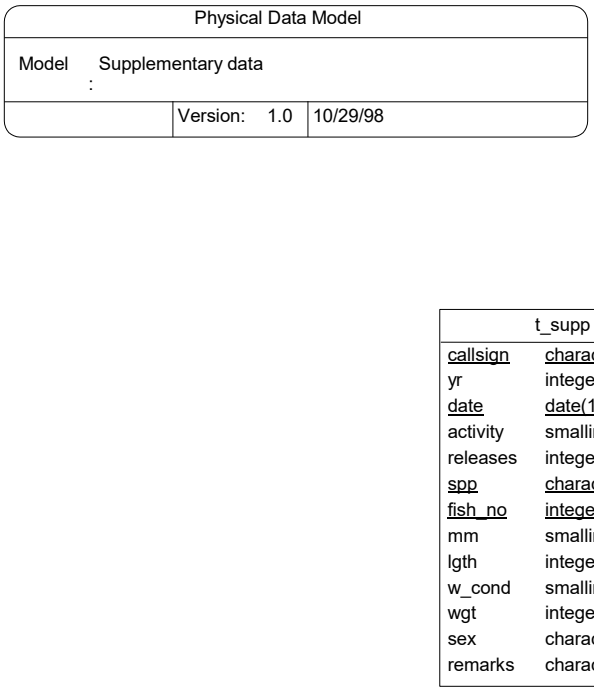

Figure 2: ERD for supplementary data within the tuna database.

Japanese data have been collected and collated into a year by month by 5x5 degree latitude and longitude summary format (Figure 3). These data are made available by the Japanese in exchange for New Zealand data under the terms of the CCSBT agreements.

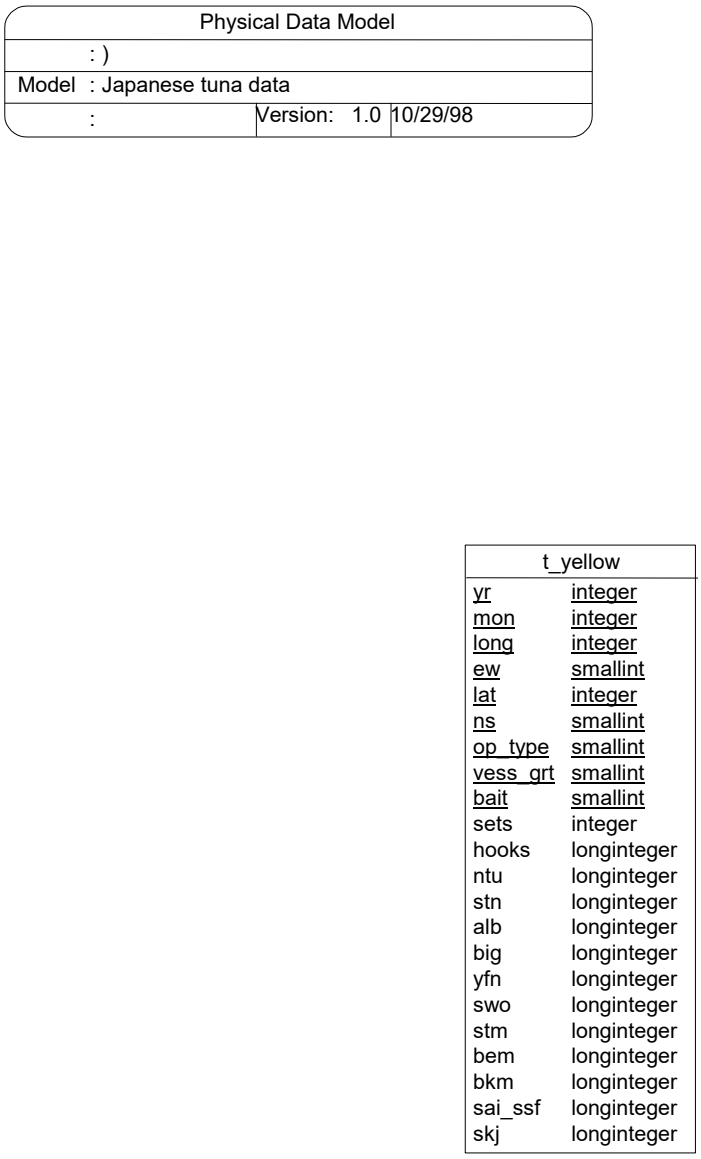

Figure 3: ERD for Japan Yellow Book data within the tuna database

4 Table summaries

The **tuna** database has 10 main tables, 9 supplementary tables.

The following is a listing and brief outline of the main tables contained in **tuna**:

- **t_meta:** contains meta information for the tuna dataset.
- **t_trip:** contains profile information on all trips.
- **t_landing:** contains landing data on a completed landing return form (CEL or CLR) for each species.
- **t_effort:** contains catch effort details on a landing return form (CEL or CLR) for each method used.
- **t_catch:** contains catch data per effort for each species.
- **t_tlset:** contains catch effort details on both old and new completed TLCER form for each long line set.
- **t_tlcatch :** contains processed weight and number for each sub catch of each tuna species caught in each set.
- **t_indcatch :** contains processed weight of each individual Southern Bluefin Tuna.
- **t_rule:** defines rules used in data grooming/loading process.
- **t_change:** contains all attribute changes in each table made during grooming/loading process.

Two supplementary tables inherited from historic tuna database are:

- **t_supp:** contains data collected from fishers in the RTMP programme.
- **t_yellow:** contains Japanese data in exchange for New Zealand under the terms of the CCSBT agreements.

Seven tables containing rejected records in grooming process, they are not shown in ERD, each table has the same structure as its corresponding table:

- **trip_rej:** trip records rejected in grooming process.
- **landing_rej:** landing records rejected in grooming process.
- **effort_rej:** effort records rejected in grooming process.
- **catch_rej:** catch records rejected in grooming process.
- **tlset_rej:** tlset records rejected in grooming process.
- **tlcatch_rej:** tlcatch records rejected in grooming process.
- **indcatch_rej:** indcatch records rejected in grooming process.

5 Tuna tables

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

5.1 t_meta

Containing data ownership information for tuna dataset, the relationships between t_meta and other tables are not enforced by foreign key, the owner_key values are assigned in data loading process.

Attribute Name	**Attribute Data Type**	**Description**
owner_key	integer	Primary key identifying owner of a dataset
owner_name	char(32)	Name of the dataset owner
project_code	char(32)	Project code used to load the dataset
subject	char(16)	Any short descriptive text for the dataset
load_date	char(3)	Date when the dataset is loaded
load_person	time(5)	Person who loads the dataset
notes	char(64)	Comments

Indexed Attribute	**Index type**	**Description**
owner_key	BTREE	Primary key

5.2 t_trip

Contains header information for each trip event associated with tuna fishing.

Attribute Name	Attribute Data Type	Description
trip_key	longinteger	Primary key using original trip key value from MFish database
version	smallint	Version of the data in MFish database
vessel_key	longinteger	MFish identification number of the fishing vessel
vessel_type	char(1)	Vessel registration type, (**D**omestic, **C**harter, **F**oreign licensed or **U**nknown)
vessel_nation	char(3)	Vessel nationality
datetime_start	time(5)	Start date and time for a trip
datetime_end	time(5)	End date and time for a trip
flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(128)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	Index type	Description
trip_key	BTREE	Primary key
vessel_key	BTREE	Normal

5.2 t_landing

Contains information for each landing event associated with tuna fishing.

Attribute Name	Attribute Data Type	Description
landing_key	longinteger	Primary key generated from a counter
version	smallint	Version of the data in MFish database
trip_key	longinteger	Foreign key to reference *t_trip*
species	char(3)	3 character species code
datetime_landing	time(5)	Date and time for the landing event
year_date	integer	Year of landing, convenient for report
month_date	smallint	Month of landing, convenient for report
day_date	smallint	Day of landing, convenient for report
form_type	char(3)	3 character code type of the form from which landing data is recorded
form_number	longinteger	Number of the form from which landing data recorded
stock	char(5)	Fish stock code
state	char(5)	Processed state of fish
bin_type	char(3)	Type of container
bin_no	integer	Number of containers
bin_weight	decimal(8,2)	Average weight of each container
green_wt	decimal(8,2)	Green weight of fish
flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(128)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	Index type	Description
landing_key	TIMESERIES	Primary key
trip_key	BTREE	Foreign key, refer *t_trip(trip_key)*

5.3 t_effort

Contains each unit of catch effort on CELR or CLR forms associated with tuna fishing.

Attribute Name	Attribute Data Type	Description
effort_key	longinteger	Primary key using event_key value of a fishing event from MFish database
version	smallint	Version of the data in MFish database
trip_key	longinteger	Foreign key to reference *t_trip*
datetime_effort	time(5)	Date and time for a catch effort event
year_date	integer	Year of fishing effort, convenient for report
month_date	smallint	Month of fishing effort, convenient for report
day_date	smallint	Day of fishing effort, convenient for report
target_sp	char(3)	Target species code
method	char(3)	Code for fishing method used
catch_total	decimal(8,2)	The total weight of catch for this fishing effort event as estimated at the time
lat	decimal(6,2)	Decimalised latitude of start of fishing truncated to 1/10th of a degree
lon	decimal(7,2)	Decimalised longitude of start of fishing truncated to 1/10th of a degree
form_type	char(3)	3 character code type of the form from which landing data is recorded
page	longinteger	Form number from which effort data recorded
sst	decimal(4,1)	Sea surface temperature
no_sets	integer	Number of sets in the day
no_hooks	integer	Total hook numbers, may vary depending on method used
effort_no	integer	Number of catch effort units, e.g. number of sets, number of lines etc depending on method used
duration	decimal(7,4)	Time spent on a catch effort
stat_area	char(5)	Statistical area in which the catch effort event started
fma	char(8)	The fisheries management area in which the fishing event started
moon_phase	decimal(4,3)	derived value from date to tell moon phase
quarter	smallint	derived value from date to tell which quarter of the year
bottom_depth	decimal(6,1)	derived value from lat/lon to tell bottom depth
soi	decimal(3,1)	derived value from date to tell southern oscillation index

flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(128)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	**Index type**	**Description**
effort_key	TIMESERIES	Primary key
trip_key	BTREE	Foreign key, refer *t_trip(trip_key)*

5.4 t_catch

Contains each estimated unit of catch recorded in CELR or CLR forms associated with tuna fishing.

Attribute Name	**Attribute Data Type**	**Description**
catch_key	longinteger	Primary key generated from a counter
version	smallint	Version of the data in MFish database
trip_key	longinteger	Redundant key for reference *t_trip*, no foreign key constraint is applied
effort_key	longinteger	Foreign key to reference *t_effort*
species	char(3)	3 character species code
catch_weight	decimal(10,2)	The weight of catch for this fishing event as estimated at the time
catch_number	decimal(10,2)	The number of catch for this fishing event as estimated at the time
flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(128)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	**Index type**	**Description**
catch_key	TIMESERIES	Primary key
effort_key	BTREE	Foreign key, refer *t_effort(key)*

5.5 t_tlset

Contains details of each tuna long line set recorded on TLCER old and new forms.

Attribute Name	Attribute Data Type	Description
tlset_key	longinteger	Primary key using dcf_key value of a fishing event from MFish database
version	smallint	Version of the data in MFish database
trip_key	longinteger	Foreign key to reference *t_trip*
vessel_key	longinteger	MFish identification number of the fishing vessel, included here to improve performance.
method	char(3)	Code for fishing method used
target_sp	char(3)	Target species
flag_newform	char(1)	Set to 'y' for new TLCER form data
page	longinteger	Form number from which effort data is recorded
datetime_set_start	time(5)	Date and time setting at longline started
lat_s	decimal(6,2)	Decimalised latitude at start of set truncated to 1/10th of a degree
lon_s	decimal(7,2)	Decimalised longitude at start of set truncated to 1/10th of a degree
fma	char(8)	The fisheries management area in which the fishing event started
stat_area	char(5)	Statistical area in which the catch effort event started
wind_spd	decimal(4,2)	Wind speed (m/s) at set start, new form only
wind_dir		Wind direction (degrees) at set start, new form only
sst	decimal(4,1)	Sea surface temperature at set start
datetime_set_end	time(5)	Date and time the longline setting is completed
lat_e	decimal(6,2)	Decimalised latitude at end of set truncated to 1/10th of a degree, new form only
lon_e	decimal(7,2)	Decimalised longitude at end of set truncated to 1/10th of a degree, new form only
cloud_type	char(5)	2 letter code for type of cloud, old form only
cloud_8ths	char(5)	Numeric code showing number of eighths cloud cover, old form only
datetime_haul_start	time(5)	Date and time at haul start
hs_wind_spd	decimal(4,2)	Wind speed (m/s) at haul start
datetime_haul_end	time(5)	Date and time at haul end

he_wind_spd	decimal(4,2)	Wind speed (m/s) at haul end
he_wind_dir		Wind direction (degrees) at haul end, new form only
he_sst	decimal(4,1)	Sea surface temperature at haul end, new form only
line_length	decimal(8,2)	Length of long line in km, old form only
line_length_nm	decimal(8,2)	Length of long line in nautical miles, new form only
no_hooks	integer	Number of hooks
no_baskets	integer	Number of baskets, old form only
no_floats	integer	Number of floats, new form only
no_light_sticks	integer	Number of light sticks, new form only
line_shooter	char(1)	'y' if line shooter is used, new form only
bait_fish_pct	smallint	Percentage of fish bait, new form only
bait_squid_pct	smallint	Percentage of squid bait, new form only
bait_artificial_pct	smallint	Percentage of artificial bait, new form only
bait_other_pct	smallint	Percentage of other bait, new form only
moon_phase	decimal(4,3)	derived value from date to tell moon phase
quarter	smallint	derived value from date to tell which quarter of the year
bottom_depth	decimal(6,1)	derived value from lat/lon to tell bottom depth
soi	decimal(3,1)	derived value from date to tell southern oscillation index
flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(256)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	**Index type**	**Description**
tlset_key	TIMESERIES	Primary key
trip_key	BTREE	Foreign key, refer *t_trip(trip_key)*

5.6 t_tlcatch

Contains catch data recorded on TLCER old and new form.

Attribute Name	Attribute Data Type	Description
tlcatch_key	longinteger	Primary key generated from a counter
version	smallint	Version of the data in MFish database
trip_key	Longinteger	Redundant key for reference *t_trip*, no foreign key constraint is applied
tlset_key	longinteger	Foreign key to reference *t_tlset*
species	char(3)	3 character species code
flag_discard	char(1)	'y' if this fish catch was discarded, new form only
state_code	char(3)	3 character code describing processed state, new form only
fish_no	Integer	Number of fish
weight	decimal(16,2)	Processed weight of the fish
flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(128)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	Index type	Description
tlcatch_key	TIMESERIES	Primary key
tlset_key	BTREE	Foreign key, refer *t_tlset(key)*

5.7 t_indcatch

Contains Southern Bluefin Tuna individual catch data recorded on TLCER old and new forms.

Attribute Name	**Attribute Data Type**	**Description**
indcatch_key	longinteger	Primary key generated from a counter
version	smallint	Version of the data in MFish database
trip_key	longinteger	Redundant key for reference *t_trip*, no foreign key constraint is applied
tlset_key	longinteger	Foreign key to reference *t_tlset*
species	char(3)	3 character species code
datetime_processed	time(5)	The start date and time for processing
weight	decimal(16,2)	Processed weight of each Southern Bluefin Tuna
flag_change	char(1)	**t** means some attribute value has been changed in grooming process, see memo field or t_change table for change details. **f** means no change.
owner_key	Integer	Refers meta record in t_meta
memo	char(128)	Comments added in loading process, mainly change info, format is rule code:attribute name:old value.

Indexed Attribute	**Index type**	**Description**
indcatch_key	TIMESERIES	Primary key
tlset_key	BTREE	Foreign key, refer *t_tlset(key)*

5.8 t_supp

Contains data collected by fishers in the RTMP programme.

Attribute Name	Attribute Data Type	Description
callsign	char(7)	Unique fisher identifier. Radio callsign
yr	integer	Trip date year
date_fishing	date(10)	Fishing date
activity	smallint	1=fishing, 2=steaming, 3=weather, 4=other
releases	integer	Number of STN released
spp	char(3)	Species code
fish_no	integer	Order lengths reported
mm	smallint	Length measurement method
lgth	integer	Length (cm)
w_cond	smallint	Process condition of weighed fish
wgt	integer	Fish weight units=kilograms
sex	char(1)	1=male 2=female 3=unknown 4=not examined
remarks	char(80)	Comment field

Indexed Attribute	Index type	Description
callsign	BTREE	Normal

5.9 t_yellow

Contains Japanese data exchanged under the terms of the CCSBT agreements.

Attribute Name	Attribute Data Type	Description
yr	integer	year
mon	integer	Month
long	integer	Longitude of the 5 degree grid
ew	smallint	Longitude code 1=East 2=West
lat	integer	Latitude of Northern (southern) border of the 5 degree grid in northern (southern) hemisphere
ns	smallint	Latitude code 1=North 2=South
op_type	smallint	Type of operation, 1=SLL
vess_grt	smallint	Vessel size code, 1=20-49 gross tonnes, 2=50-99, 3=100-199, 4=200 plus
bait	smallint	Bait code, 1=mackerel, 2=squid, 3=others & mixed
sets	integer	Number of sets
hooks	longinteger	Number of hooks
ntu	longinteger	Number of bluefin caught
stn	longinteger	Number of southern bluefin caught
alb	longinteger	Number of albacore caught
big	longinteger	Number of bigeye caught
yfn	longinteger	Number of yellowfin caught
swo	longinteger	Number of broadbill swordfish caught
stm	longinteger	Number of striped marlin caught
bem	longinteger	Number of blue marlin caught
bkm	longinteger	Number of black marlin caught
sai_ssf	longinteger	Number of sailfish & shortbill spearfish caught
skj	longinteger	Number of skipjack caught

6 Data Loading and Validation

A set of loading and validating shell scripts are used to groom and validate tuna fishing data extracted from the MFish database. Because the amount of data involved, each run of these scripts can only efficiently process a relatively small portion of the data by giving a date window; e.g., several months or a year. The detailed rules used in the loading process are listed in Appendix A.

7 Acknowledgements

The author would like to thank T.E Murray and Lynda Griggs for writing and reviewing the business rules applied to tuna data, Dave Banks and Dave Fisher for review and editorial comment, and Kevin MacKay for his technical input for this document. Part of the document is based on H.A. Dean's original tuna database document.

8 References

Dean, H.A. 1998: Database documentation: tuna. *NIWA Internal Report No. 37*. 27p.

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

.

A Tuna Database Processing Rules

The rule codes are underlined; they will appear in the memo field and the t_change table. The rule description will be contained in the t_rule table. Rules not underlined will not appear in the tuna database, they are implemented in the design.

A.1 General Rules

GR1. Each rejected record will be kept in a reject table.
G2. Each table will have a change flag field and a memo field, whenever its attribute value is changed, the original value and the rule code applied will be written into the memo field, and the flag set to “T”.
G3. Each attribute value to be changed will also be kept in a change table with the rule applied and its old and new values.
G4. The original data set will be dumped out as comma separated plain text files automatically by the loading program. As the data structure may be different from that of the *tuna* database, the heading is included in each file.

A.2 Data Scope Rules

The following subset of rules is to be used to extract related data from MFish’s WAREHOU database.
DS1. All data set should satisfy the listed general criteria

- Related trip/effort/landing/processed date is between the specified date range.
- Use interpreted version of data.

DS2. The bulleted species codes and method codes are used when appropriate

- Used species codes are SPP=('ABO','ALB','BIG','BEM','BKM','BTU','DSM','FTU','MAR','NTU','SAI','SKJ','SSF','STM','STN','STU','SWO','TUN','YFN')
- Used method codes are METHOD=('T', 'TL', 'PL', 'HL', 'SLL', 'PS')

DS3. Trip data

- Data source: x_trip_event
- Selection criteria: form_type is in ('CEL','TUN','CLR')
- Remove those trips un-relevant to tuna-fishing.

DS4. Landing data

- Data source: x_landing
- Selection criteria: form_type in ('CEL', 'CLR') AND species_code in SPP

DS5. CELR Effort data

- Data source: x_fishing_event
- Selection criteria: form_type='CEL' AND (primary_method in METHOD OR target_species in SPP)

DS6. CELR Catch data

- Data source: x_estimated_subcatch
- Selection criteria: form_type='CEL' AND (primary_method in METHOD OR target_species in SPP)

DS7. Longline Effort data

- Data source: x_fishing_event f, x_environment_data e and x_bait
- Selection criteria: form_type='TUN'

DS8. Longline Individual Catch data

- Data source: x_tuna_individual_catch
- Selection criteria: form_type='TUN' AND specprod_action_type='SIN'

DS9. Longline Catch data

- Data source: x_processed_catch

- Selection criteria: form_type='TUN'

A.3 Data Relationship Rules

GAP: If a record has a parent key which can not be found in its parent table, then move the record into reject table.
DUPLICATE: If two or more records have the same primary key, move those having lower version number or having same version number but higher record number into reject table.
ORPHAN: If a record has null parent key then the record is regarded as an orphan record.
ADOPT: If an orphan record has the same vessel as that of a parent record and its event date is between start event date and end event date of the parent record, then this parent record is regarded as the parent of the orphan record.
PSEUDO: For each vessel in the orphan records which can not be adopted, create a pseudo trip which has minimum fishing date in a month as trip start date and maximum fishing date in the same month as trip end date, and uses number greater than 2000000000 as trip_key.

A.4 Position Rules

PR1. Statistical Area

STAT: Set stat area to null if stat area is not a valid stat area, by valid we mean the stat areas in the General Statistical Areas.

PR2. Latitude and Longitude

POSERR: For a trip having more than or equal to 5 fishing positions (sets), firstly calculate average travelling speed and average travelling distances between each adjacent fishing positions. If both differences between a position's adjacent distances and its average travelling distances are more than 3 degrees, this position is thought to be wrong. If only one erroneous position found then use the middle position between the adjacent ones as correct position, otherwise mark the position as POSCHK1 in the memo field.
LAT: Set latitude to null if not between -25 to -60 degrees if stat area is valid.
LON: Set longitude to null if not between 160E to 170W degrees if stat area is valid.
POSCHK1: For those position errors found in rule POSERR but can not be fixed.
POSCHK2: If a position is not inside its stat area, flag the position as questionable but does nothing to fix.
CENTROID: Set latitude and longitude to that of centroid of stat area if latitude or longitude is null.

PR3. FMA

Set FMA according to latitude and longitude for start fishing position.
FMA: if FMA is 'NI' or 'SI' then set the FMA to null, if stat area is available then set latitude and longitude to that of centroid of the stat area, otherwise set to null.

A.5 Vessel Rules

VSLDUM Use master vessel table (t_vessel_tuna in vessel database) to fill vessel type and/or nationality fields for nulls or unknown value 'U' and 'DUM'.
VSLONE: If there is only one value for vessel type of a vessel except 'U', then set that known vessel type to all the 'U' vessel type.
This rule also applies to vessel nationality, set the known nationality to 'DUM' value.
VSLD: For vessels having vessel_nation as 'NZL' but vessel_type not 'D', set vessel_type to 'D'.
VSLNZL: For vessels having vessel_type as 'D' and vessel_name as 'DUM', set vessel_nation to 'NZL'.
VSLC: For vessels having vessel_nation not as 'NZL' and 'DUM' but vessel_type as 'D', set vessel_type to 'C'.
VSLNEW: For new vessel keys not found in master vessel table, update the master vessel table. This rule code is recorded in t_vessel_tuna.memo field.
VSLCHG: For discrepancy of vessel type or nationality for the same vessel between that in master vessel table and in tuna trip table, treat it as normal change, and insert a new record in the master vessel table with its memo field marked with this rule code.

A.6 Other Rules

OR1. Tlset Date Time

SETEND: If datetime_set_end is before datetime_set_start, set it to null.

HAULSTART: If datetime_haul_start is before datetime_set_end or datetime_set_start, set it to null.
HAULEND: If datetime_haul_end is before datetime_haul_start, set it to null.

OR2. Sea Surface Temperature

SST: Set sea surface temperature to null if it's not between 10 to 30 degrees.

OR3. Species Code

SPNUL: Set species code to 'UNI' if species code is null.
SPUNI1: Re-assign species code to "UNI" if the species code is within the following list:
'AGR','BBA','BCO','BEN','BUT','CDL','DIS','DOS','ESQ','FBA','FRO','GSC','HAK','LEA','LIN','MAC','MDO',
'MIX','MOK','OFF','ORH','OTH','PIF','PMA','POP','RBY','RCO','RDO','REP','RSN','RUB','SEO','SKA','SKI',
'SLR','SPE','SQU','SSK','STA','SWA','TAR'.
SPUNI2: Re-assign species code to 'UNI' if the species code is within the following list:
'BMA','EMA','KAH','SNA','TRE' AND

- For all longline catch effort records.
- For landing/CELR effort and catch records where method is not 'PS' (Purse Seining).

SPSHA: Reassign species code to 'SHA' if the species code is within the following list:
'BSH','GSH','HHS','SEV','SOP','SPD'.
SPOFH: Reassign species code to 'OFH' if the species code is 'OIL'.
SPPOS: Reassign species code to 'POS' if the species code is 'POR'.
SPDAS: Reassign species code to 'DAS' if the species code is 'RAY'.
SPSHF: Reassign species code to 'SHF' if the species code is 'SFN'.

A.7 Catch Effort Rules

The average catch weight per trip for CELR in the following text means "green weight / estimated catch number".

CE1 Green Weight

GWCAL (Green Weight Calculation): If green weight is null use bin number X bin weight to get the green weight.

Fishing Method

MTHSLL: Set method to 'SLL' if fishing method for longline effort is null.

CE2. Fishing Effort

SETNO (new) : For method 'PS', set no_sets to 1 in the following conditions: 1.the values in no_sets column per day per vessel are null; 2. the values in no_sets column per day per vessel are sequential numbers; 3. the values in no_sets column per day per vessel are all the same as the number of fishing efforts.
PSSET7 (BR2): For method 'PS', if sum of no_sets per day per vessel > 7, set no_sets to null.
LONSET2 (BR17): For method 'SLL'/'BLL' if number of sets per day > 2, set number of sets to null.
50HK4000 (BR18): For method 'SLL'/'BLL' if number of hooks per set is not between 50 and 4000, set number of hooks to null.

CE3. Catch

NUMWGT (BR20ab): If total catch number per trip per species > 1000 or if no green weight is available and average catch number per species per record > 100, set the catch number to catch weight except ALB, SKJ and YFN species.
NUMWGT2(new) If the average catch weight fails the lower limit (2kg or more) of the following rules and it is between 0.7 to 1.3 (the difference of estimated catch number and green weight is close to <= 30%), then set the estimated catch number to weight.
1PS160(BR1): For method 'PS' and vessel overall length <= 50m if estimated catch weight per set > 160t or < 1t, set the catch to null.
1PS350 (new): For method 'PS' and vessel overall length > 50m if estimated catch weight per set > 350t or < 1t, set the catch to null.
ALB300 (BR3): For method 'T'/'PL' and species 'ALB' if catch number per day > 300, set catch number to weight.
ALB10 (BR5): For method 'T/PL' and species 'ALB' if green weight per trip > 10t, set green weight to null.
2ALB20(BR6): For method 'T/PL' and species 'ALB' if average catch weight <2 kg or > 20 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null.
10STN225(BR8): For method 'T/PL' and species 'STN' if average catch weight < 10 kg or > 225 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null.
25NTUTOR350(BR9): For method 'T/PL' and species in ('NTU', 'TOR') if average catch weight < 25 kg or

> 350 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null.

1YFN70(BR10): For method ‘T/PL’ and species ‘YFN’ if average catch weight < 1.2 kg or > 70 kg, set the catch number to null.
2ALB45 (BR11): For method in (‘HL’, ‘SLL’, ‘BLL’,’DL’,’TL’) and species ‘ALB’ if average catch weight < 2 kg or > 45 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null; for TLCER form set catch to null.
15BIG210(BR12): For method in (‘HL’, ‘SLL’, ‘BLL’,’DL’,’TL’) and species ‘BIG’ if average catch weight < 15 kg or > 210 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null; for TLCER form set catch to null.
25NTUTOR350L(BR13): For method in (‘HL’, ‘SLL’, ‘BLL’,’DL’,’TL’) and species in (‘NTU’,’TOR’) if average catch weight < 25 kg or > 350 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null; for TLCER form set catch to null.
10STN250 (BR15): For method in (‘HL’, ‘SLL’, ‘BLL’,’DL’,’TL’) and species ‘STN’ if average catch weight < 10 kg or > 250 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null; for TLCER form set catch to null.
7SWO650 (BR16): For method in (‘HL’, ‘SLL’, ‘BLL’,’DL’,’TL’) and species ‘SWO’ if average catch weight < 7 kg or > 650 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null; for TLCER form set catch to null.
10YFN200 (BR14): For method in (‘HL’, ‘SLL’, ‘BLL’,’DL’,’TL’) and species ‘YFN’ if average catch weight < 10 kg or > 200 kg, firstly apply rule NUMWGT2, if not applicable then set the catch number to null; for TLCER form set catch to null.
SKJ3 (BR19): For method ‘PS’ and species ‘SKJ’ if the ratio of green weight over estimated catch weight > 3 set the catch to null.