Database documentation for the Ministry for Primary Industries tagging database

tag

K. A. Mackay & B. A. Wood

NIWA Fisheries Data Management
Database Document Series

Updated June 2015

Contents

List of Figures

Revision History

Version	Change	Date	Responsible
1.0	First release	1993	Brent Wood
1.1	Revised as NIWA Internal Report No. 72	2000	K.A Mackay & B.A. Wood
1.2	Revision	5 March 2002	Kevin Mackay
1.3	Added Revision History table, changed comment on t_release.weight to kg from g, D Fisher's instruction.	28 August 2003	Fred Wei
1.4	Updated the document to reflect the length of some previously increased character data types, and existing but unlisted additional attributes.	3 March 2004	David Fisher
1.5	Altered proj_code to char(16), dist to decimal(8,3).	5 August 2004	Fred Wei
1.6	Altered t_lfreq.station_code to char(12).	8 November 2004	Fred Wei
2.0	Huge change, redesigned, 9 new tables added	28 Jun 2007	Fred Wei
2.1	Added colour char(16,1) in t_tag_batch	27 August 2008	Fred Wei
2.2	replaced type_code with tag_type	18 November 2008	Fred Wei
2.3	Minor editorial corrections, including to section 4.	17 July 2009	David Fisher
2.4	Added rel6, rec6. extended rel2, rec2, fisher_name	8 Jun 2010	Fred Wei
2.5	Change t_link.fish_id to tag_no	17 Feb 2012	Fred Wei
3.1	Postgres version	June 2015	D Fisher, F Wei

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 tag survey database **tag**, and is a part of the database documentation series produced by NIWA. It supersedes the original documentation by Wood (1993) 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. The ERD graphically shows how all the tables fit in together, and their relationships to other databases. This database has been implemented as a schema within the Postgres database called **fish**.

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

2. Tagging Programmes

2.1 Sources of tagging data

Tagging programmes have been used to provide information on fish and fisheries in New Zealand for many years. Many of these programmes are described in a variety of publications, with summaries of tagging programmes carried out in New Zealand included in Crossland (1982) and Murray (1990). A wide variety of species have been the subject of such studies, including finfish, shellfish and rock lobsters.

To facilitate the storage and access of data collected by a wide range of tagging programmes requires a flexible database structure. This is likely to be more complex than that required for any single programme. A brief overview of some different programmes follows to help illustrate this:

The 1980/1981 kahawai programme involved a single target species. The animals were marked and released throughout New Zealand during the two year period. Many were measured when tagged, and samples of animals were also taken to provide additional age/growth information. Only one tag was applied to each animal. Animals were not injected with any biochemical markers. This is an example of a very straightforward tagging programme, intended to provide information on movement, age/growth and relative levels of effort by method and fisher type (recreational/commercial).

The co-operative gamefish tagging program is an ongoing tagging programme initially run by MAF Fisheries in co-operation with the International Game Fishing Association. This programme has several target species, and it is generally not possible to get an accurate length or weight of the animals tagged. This programme is particularly unusual in that there is no single target species.

The bluenose tagging programme run off the Wairarapa coast in 1987 does have a single target species. To tag bluenose without damaging them it is necessary to apply the tag to the animals at the depths they are normally found, generally over 200m deep. To do this, baited hook tags attached to lines set in appropriate areas were used. Of the baits/tags which were taken from the lines, it is not possible to say what species (or if the sea bed) "took" the tags. The actual species tagged cannot be known, except for those tags which were recaptured.

A 1987 snapper tagging programme in Tasman Bay had every tenth fish double tagged to help determine the level of tag shedding which occurred. The fish were also injected with tetracycline to assist with age determination upon recapture.

Another snapper tagging programme, this time in the Hauraki Gulf during 1994, had snapper tagged with coded wire tags. All landed snapper within a factory were passed through an electronic scanner to detect the tagged fish. Tag numbers were only known at the time of detection, but not release. A known number of snapper were seeded with tags at the factory to calculate the detection rate of the electronic scanner.

2.2 Data loading and validation

As the data from different tagging programmes has been stored in a variety of formats prior to the establishment of the **tag** database, no standard system has been developed for loading data into the database. Many of the validation rules also vary between tagging programmes so these also are not implemented on the database as a whole. Prior to data being loading to the database, for each tagging programme, appropriate validation rules should be developed and the data checked against these rules.

The referential and range checks listed in the table structures and shown in the Entity Relationship Diagram are the only validation checks enforced.

3. Data structures

3.1 Table relationships

This database contains several tables. The ERD for **tag** (Figure 1) shows the physical data model structure[1] 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 has been represented 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.

Some of the tables in the **tag** database have some attributes, called foreign keys[3], which contain standard NIWA fisheries codes, such as *species* and *stage_meth*. These attributes provide links to tables in the **rdb** (research database) database, which contains the definitive list of standard codes. Therefore, an expanded ERD for these tables will follow (Figure 2).

Section 5 shows a listing of all the **tag** tables as implemented by the Postgres DBMS. 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*])

where the attribute(s) make up the primary key and the index name is the primary key name. Note that the typographical convention for the above (and subsequent) format is the square brackets [ ] may contain an item that is repeated zero or more times.

A primary key prevents records with duplicate key values from being inserted into the table, e.g., a new t_project record with an existing proj_id, and hence ensures that every record can be uniquely identified.

---

1 Also known as a database *schema*.
2 A primary key is an attribute or a combination of attributes that contains an unqiue 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.

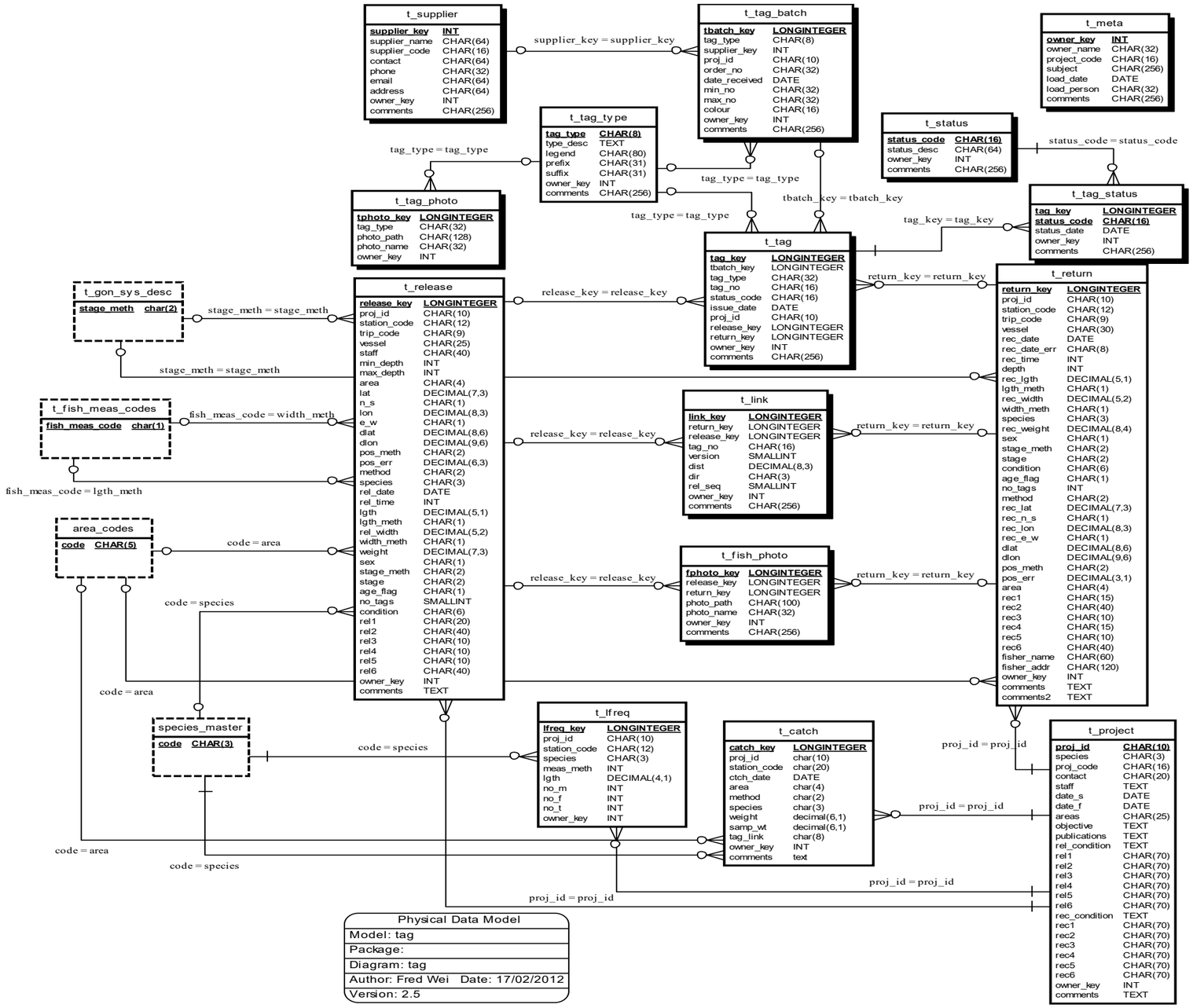
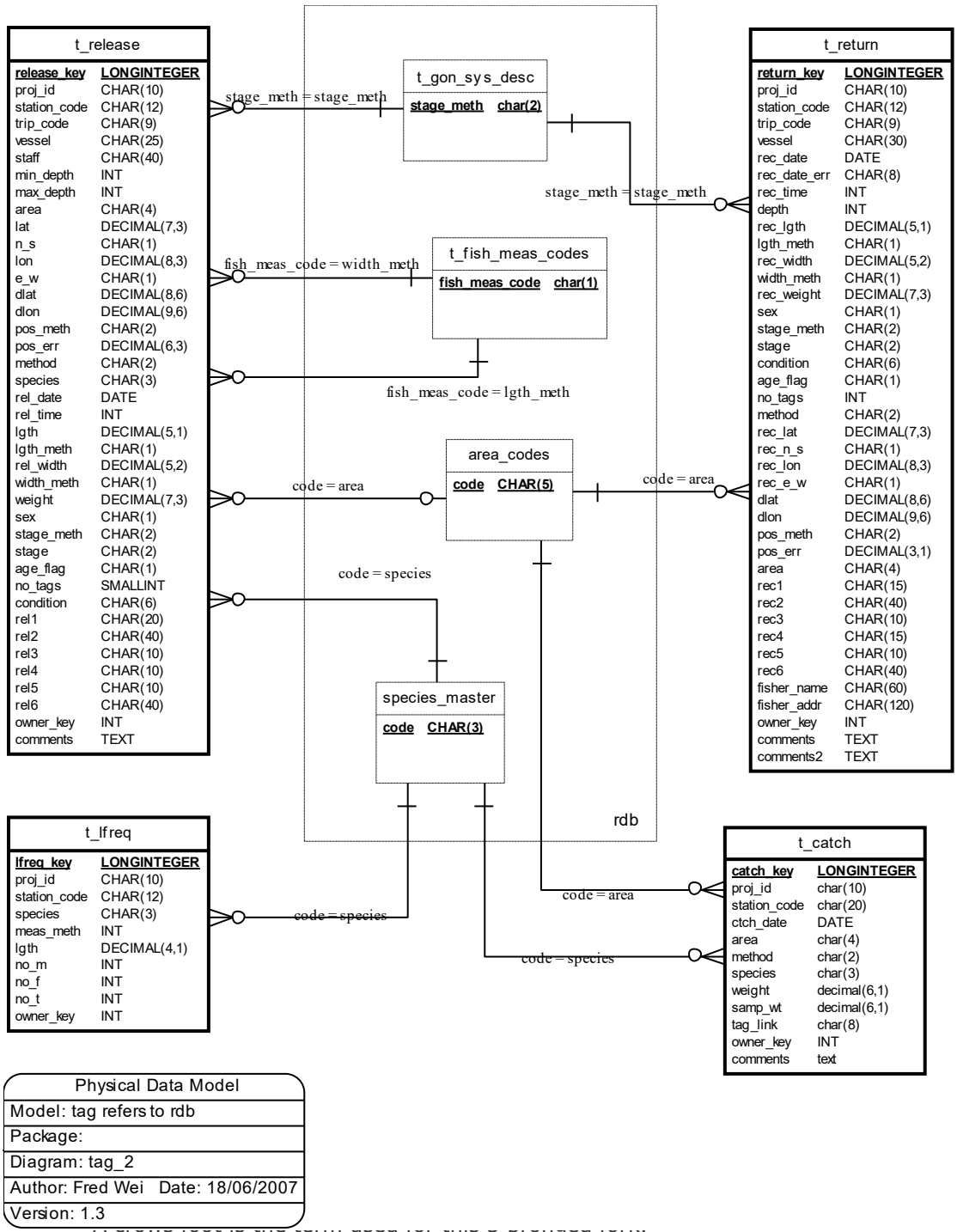


Figure 1 : Entity Relationship Diagram (ERD) of the **tag** database.

The **tag** database is implemented as a relational database. That is, each table is a special case of a mathematical construct known as a *relation* and hence elementary relation theory is used to deal with the data within tables and their relationships between them. All relationships in **tag** are of the type *one-to-many*[4]. This is shown in the ERD by connecting a crows foot[5] (indicating 'many') from the child table (e.g., *t_catch*) to the parent table (e.g., *t_project*) with a single line (indicating 'one') pointing to the parent.

Figure 2 : Expanded ERD of **tag** tables showing relationships to **rdb** tables.



[5] A crows foot is the term used for this 3-pronged fork.

Every relationship has a mandatory or optional aspect to it. That is, if a relationship is mandatory, then it has to occur at least once, while an optional relationship might not occur at all. For example, in Figure 1, consider that relationship between the table *t_project* and it's child table *t_catch*. The symbol "O" by the child *t_catch* means that a tag project can have zero or many catch records, while the bar by the parent *t_project* means that for every catch record there must be a matching tag project record.

Most of these tables contain foreign key constraints[6] , which link these tables to each other and to tables in the **rdb** database (Figure 2). 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: a parent record is deleted; the parent record is altered so that the relationship is lost; or a child record is entered without a parent record. Foreign key constraints are shown in the table listings by the following format:

```
Foreign-key constraints:
    "foreign key name" FOREIGN KEY (attribute[,attribute])
    REFERENCES parent table (attribute[, attribute])
```

For example, consider the following constraint found in the table *t_release*:

Foreign-key constraints:
"fk_t_release_t_project" FOREIGN KEY (proj_id) REFERENCES project(proj_id)

This means that the value of the attribute proj_id in a *t_release* record must already exist in the parent table *t_project* or the record will be rejected and an error message will be displayed.

A delete constraint implies that for a record to be deleted from a table, the values of the constrained attributes must not be equal to the values of the corresponding attributes in any record of the constraining table. This is used to prevent a parent record from being deleted while child records still exist.
Foreign key constraints enforce both insert and delete constraints.

All tables in this 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 on attributes that are not primary or foreign keys are listed using the following format:

Indices: index_name btree (*attribute*[, *attribute*])

Note that indices may be simple, pointing to one attribute or composite pointing to more than one attribute.

[6] Also known as integrity checks.

3.2 Database design

There are four main subject areas of the tag data model which are described below, summarized by database table:

3.2.1 Meta data

t_meta: defines dataset ownership with each table having a mandatory (not null) owner_key attribute which is populated in the data loading process. This table also records the date when each dataset is loaded to the database.
t_project: stores information describing a tagging programme undertaken. This information does not necessarily relate to data held in other tables in the database, which enables this table to be used as a reference to tagging programmes which have been carried out, even if the actual tagging data is not stored in the database. This table is uniquely keyed on the *proj_id* attribute.

3.2.2 Tag data

t_tag: records individual tags used or to be used in a tagging programme.
t_supplier: contains information about tag suppliers.
t_tag_batch: contains information about batches or orders of tags.
t_tag_type: contains description of each type of tag, along with a code representing the tag type.
t_tag_photo: records the whereabouts of files of tag photos.
t_status: contains status information of tags, eg ordered, received, released.
t_tag_status: each tag may have a different status, this table links each tag with its status to the status table.

3.2.3 Fish data

t_release: relates to actual animals tagged and includes associated location, time, gear method information etc. Each record represents one animal being marked and released. It is linked to the project table by the *proj_id* attribute. To enable programme specific data to be stored in a generic database, five user defined attributes have been implemented. These can each hold up between ten and twenty characters and are used if required. The data stored in these attributes is described in the appropriate attributes in the project table.

t_return: contains information pertaining to tagged animals that have been recaptured. This is linked to the project table directly via the *proj_id* attribute. This table also has five user defined attributes similar to those in the release table, but for programme specific recapture or return data.
t_link: establishes links between a release event and return event.
t_fish_photo: records whereabouts of files of fish photos.

3.2.4 Catch data

Tagging projects are sometimes carried out in conjunction with other programmes including trawl surveys and market or catch sampling programmes. In these cases,

catch and length frequency data would be held in databases such as **trawl**, **market**, **scallop**, **rlcs**, etc. Where there is no appropriate database to store associated catch and or length frequency data these data are kept in the *t_catch* and *t_lfreq* tables respectively.

t_lfreq: holds length frequency data collected on catches from which animals were tagged, as collected by some programmes. Length frequency data is linked to the *t_project* table by the *proj_id* attribute, and may be linked to the *t_release*, *t_return,* or *t_catch* tables by the *proj_id* and *station_code* attributes.

t_catch: holds the total catch weight of catches from which animals were tagged or recaptured as recorded by some tagging programmes. These data are linked to the *t_project* table by the *proj_id* attribute, and may be linked to either the *t_release* or *t_return* tables by the *proj_id* and *station_code* attributes.

3.3 Handling orphan tag return records

In strictly logical terms, a tag return record can not exist without a matching tag release. However, the reality is that tags get damaged resulting in tag numbers becoming partially or wholly illegible and impossible to be matched against any one release record. In such instances, dummy release records may be inserted into the *t_release* table to match the damaged returned tags.

3.4 Multiple releases and returns

With certain species, such as crayfish, individual tagged animals may be released and recaptured many times. In such cases, one animal is represented by multiple records in both the *t_release* and *t_return* tables.

Care must be taken when joining *t_release* and *t_return* using the *proj_id*, *tag_no* key*,* as these cases result in a many-to-many relationship.

However, one could resolve this issue by using one of the user-defined fields for a sequential release and return number. Each time an animal is released, the release number is incremented by one (first release = 1). Similarly, each time the animal is recaptured, the return number is also incremented by one. The keys needed to match a tag return with its appropriate release are then: *proj_id*, *tag_no* and release/return number. In practice this sequential release and return number was seldom if ever implemented. Issues with duplicate tag numbers also confound a small number of datasets, and these duplicate tag numbers result in a many-to-many relationship.

This revision of the tag database (Version 2.0) incorporates a new table *t_link,* which records the link or association between the *t_return* and *t_release* tables.
This table includes a *rel_seq* attribute to record the chronological sequence of each release. This table was not back populated when it was created in June 2007 because experienced interpretation of individual records is required in some cases to generate the correct links.

3.5 Tagging programme requirements and user-defined fields

By in large, the attributes of the main tag tables (*t_release* and *t_return*) are for the main data items that are common for the majority of tagging programmes. However, each programme has certain data fields that are relevant for that specific programme only. Rather than constantly add attributes to these table when the need arises, this database was created with the flexibility of user-defined fields (*rel1* - *rel5* and *rec1* - *rec5* in the *t_release* and *t_return* tables respectively) that will hold any kind of data. Interpretation of these fields will differ between programmes, their usage's are defined in the *t_project* table.

4. Table summaries

The **tag** database has fifteen tables containing tag data. The following is a listing and brief outline of the tables contained in **tag**:

1. **t_meta :** contains dataset meta-data information.
2. **t_project :** contains details and descriptions of individual tagging projects, including definitions of user-defined fields used in the *t_release* and *t_return* tables.
3. **t_release :** contains details of tagged animal releases.
4. **t_return :** contains details of tagged animal returns or recaptures.
5. **t_tag :** contains information of individual tags used or to be used in a tagging program. If a tag is released or returned more than once, then t_tag_status where populated will contain status information for each release and return event.
6. **t_link :** contains interpretations of release and return data.
7. **t_tag_batch :** contains tag batch information, each order is typically treated as a batch.
8. **t_tag_type :** contains descriptions of the different types of tags.
9. **t_status :** contains detailed tag status information.
10. **t_tag_status :** contains detailed tag status information for each release or return for each tag.
11. **t_supplier :** contains information about tag suppliers.
12. **t_tag_photo :** contains locations of tag photos.
13. **t_fish_photo :** contains locations of tagged animal photos.
14. **t_lfreq :** contains length frequency data for some tagging projects.
15. **t_catch :** contains catch data for some tagging projects.

5. tag tables

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

5.1 Table 1: t_meta

```
Comment: Table to contain data set ownership information, the
         relationships between t_meta and other tables are not
         enforced by foreign key, the owner_key values are
         assigned in data loading process.
```

Column	Type	Null?	Description
owner_key	integer	No	Primary key to identify owner of a dataset.
owner_name	character varying(32)		Name of the dataset owner.
project_code	character varying(16)		Project code associated with the dataset or used to load the dataset.
subject	character varying(256)		Any short descriptive text for the dataset.
load_date	date		Date when the dataset is loaded.
load_person	character varying(32)		Person who loads the dataset.
comments	character varying(256)		Text comment on this dataset.

```
Indexes:
   "pk_t_meta" PRIMARY KEY, btree (owner_key)
```

5.2 Table 2: t_project

Comment: Table to hold information relating to individual tagging programmes.

Column	Type	Null?	Description
proj_id	character varying(10)	No	Primary key to distinguish different tagging programmes.
species	character varying(3)	No	The species code for mono-specific tagging programmes, otherwise 'MIX'.
proj_code	character varying(16)		Project code for the initial tagging programme (if available).
contact	character varying(20)	No	The staff member who is the main contact regarding the data.
staff	text		Staff members involved with the tagging programme.
date_s	date		Start date of the tagging programme.
date_f	date		Finish date of the tagging programme.
areas	character varying(25)	No	Area codes as found in rdb.area_codes.
objective	text		The goals/objectives of the programme.
publications	text		Lists publications used to plan or describing data from the programme.
rel_condition	text		Description of the usage of the t_release condition field.
rel1	character varying(70)	No	Description of the usage of the t_release.rel1 field. ('not used' if it isn't.)
rel2	character varying(70)	No	As for rel1.

rel3	character varying(70)	No	As for rel1.
rel4	character varying(70)	No	As for rel1.
rel5	character varying(70)	No	As for re11.
rel6	character varying(70)		As for re11.
rec_condition	text		Description of the usage of the t_returns condition field.
rec1	character varying(70)	No	Description of the usage of the t_return.rec1 field. ('not used' if it isn't.)
rec2	character varying(70)	No	As for rec1.
rec3	character varying(70)	No	As for rec1.
rec4	character varying(70)	No	As for rec1.
rec5	character varying(70)	No	As for rec1.
rec6	character varying(70)		As for rec1.
owner_key	integer	No	Refer to meta record of the dataset.
comments	text		Any text comments to be made on the tagging programme.

Indexes:
"pk_t_project" PRIMARY KEY, btree (proj_id)

5.3 Table 3: t_release

Comment: Table to contain information on individual animals released.

Column	Type	Null?	Description
release_key	integer	No	Primary key to identify a release.
proj_id	character varying(10)	No	Foreign key to refer to a tagging programme.
station_code	character varying(12)		Station (or "release site") identifier, incorporating the trip identifier.
trip_code	character varying(9)		Optional trip code identifier.
vessel	character varying(25)		Name of vessel used to capture the animals for tagging.
staff	character varying(80)		Staff involved in this release.
min_depth	integer		Generally either bottom or gear depth as appropriate.
max_depth	integer		Generally either bottom or gear depth as appropriate.
area	character varying(5)		Area code from rdb.area_codes where release occurred.
lat	numeric(7,3)		Latitude (as DDMM.mmm) where release occurred.
n_s	character varying(1)		Release latitude North or South of equator.
lon	numeric(8,3)		Longitude (as DDDMM.mmm) where release occurred.
e_w	character varying(1)		Release longitude East or West
dlat	numeric(8,6)		Latitude (as decimal degree) where release occurred.

dlon	numeric(9,6)	Longitude east of Greenwich (as decimal degree) where release occurred.
pos_meth	character varying(2)	2 character code for the method of fixing the position. Refer rdb.t_fix_meth_codes.
pos_err	numeric(6,3)	Radius of margin of error of the position (nautical miles)
method	character varying(2)	Method used to capture the animals to be tagged.
species	character(3)	3 character species code.
rel_date	date	Date when release occurred.
rel_time	integer	Time of day (24hr) when release occurred.
lgth	numeric(5,1)	Length of animal tagged (cm to 1 decimal)
lgth_meth	character varying(1)	1 character fish length measurement type code. Refer rdb.t_fish_meas_codes.
rel_width	numeric(5,2)	Width of animal tagged (cm to 1 decimal)
width_meth	character varying(1)	1 character fish width measurement type code. Refer rdb.t_fish_meas_codes.
weight	numeric(7,3)	Weight of animal tagged (may be an estimate) (kg)
sex	character varying(1)	Sex code: null = not known, 1 = male, 2 = female, 3 = indeterminate or immature, 4 = did not sex.
stage_meth	character varying(2)	2 character code to describe gonad staging method. Refer rdb.t_gon_sys_desc.

stage	character varying(2)		Stage of sexual maturity (codes vary by species).
age_flag	character varying(1)		Flag whether aging material was taken.
no_tags	smallint		Number of tags attached to the animal.
condition	character varying(6)		Physical condition of released animal.
rel1	character varying(20)		User defined field, as defined in project table.
rel2	character varying(40)		User defined field, as defined in project table.
rel3	character varying(10)		User defined field, as defined in project table.
rel4	character varying(10)		User defined field, as defined in project table.
rel5	character varying(10)		User defined field, as defined in project table.
rel6	character varying(40)		User defined field, as defined in project table.
owner_key	integer	No	Refer to meta record of the dataset.
comments	text		Any text comments on the release site.
position	geometry		Position of vessel as gis point type.

```
Indexes:
   "pk_t_release" PRIMARY KEY, btree (release_key)
   "nx_t_release_position" gist ("position")
Check constraints:
   "enforce_dims_position" CHECK (ndims("position") = 2)
   "enforce_geotype_position" CHECK (geometrytype("position") =
    'POINT'::text OR "position" IS NULL)
   "enforce_srid_position" CHECK (srid("position") = 4326)

Foreign-key constraints:
```

```
"fk_t_release_area_codes" FOREIGN KEY (area)
 REFERENCES rdb.area_codes(code)
"fk_t_release_species_master" FOREIGN KEY (species)
 REFERENCES rdb.species_master(code)
"fk_t_release_t_fish_meas_codes" FOREIGN KEY (lgth_meth)
 REFERENCES rdb.t_fish_meas_codes(fish_meas_code)
"fk_t_release_t_fish_meas_codes_2" FOREIGN KEY (width_meth)
 REFERENCES rdb.t_fish_meas_codes(fish_meas_code)
"fk_t_release_t_gon_sys_desc" FOREIGN KEY (stage_meth)
 REFERENCES rdb.t_gon_sys_desc(stage_meth)
"fk_t_release_t_project" FOREIGN KEY (proj_id)
 REFERENCES tag.t_project(proj_id)
```

5.4 Table 4: t_return

Comment: Table to hold information describing recaptured animals or returned tags.

Column	Type	Null?	Description
return_key	integer	No	Primary key to identify a recapture event.
proj_id	character varying(10)	No	Foreign key to refer to a tagging programme.
station_code	character varying(12)		Station or recapture site identifier, incorporating the trip identifier.
trip_code	character varying(9)		Optional trip code identifier.
vessel	character varying(30)		Name of vessel where the recapture occurred.
rec_date	date		Date the animal was recaptured.
rec_date_err	character varying(8)		Describes the "error" in the assigned rec_date.
rec_time	integer		Time of day (24hr) recapture took place.
depth	integer		Depth of water OR gear where recapture occurred.
rec_lgth	numeric(5,1)		Length of animal tagged (cm to 1 decimal).
lgth_meth	character varying(1)		1 character fish length measurement type code. Refer rdb.t_fish_meas_codes.
rec_width	numeric(5,2)		Width of animal tagged (cm to 1 decimal).
width_meth	character varying(1)		1 character fish width measurement type code. Refer rdb.t_fish_meas_codes.
species	character(3)		3 char species code

rec_weight	numeric(8,4)	Weight of animal tagged (may be an estimate) (kg).
sex	character varying(1)	Sex code: null = not known, 1 = male, 2 = female, 3 = indeterminate or immature, 4 = did not sex.
stage_meth	character varying(2)	2 character code to describe gonad staging method. Refer rdb.t_gon_sys_desc.
stage	character varying(2)	Stage of sexual maturity (codes vary by species).
condition	character varying(6)	Physical condition of returned animal.
age_flag	character varying(1)	Flag whether aging material was taken.
no_tags	integer	Number of tags attached to the animal.
method	character varying(2)	Method used to capture the animals to be tagged.
rec_lat	numeric(7,3)	Latitude (as DDMM.mmm) where animal was recaptured.
rec_n_s	character varying(1)	Recapture latitude North or South of Equator.
rec_lon	numeric(8,3)	Longitude (as DDDMM.mmm) where animal was recaptured.
rec_e_w	character varying(1)	Recapture longitude East or West.
dlat	numeric(8,6)	Latitude (as decimal degree) where animal was recaptured.
dlon	numeric(9,6)	Longitude east of Greenwich (as decimal degree) where animal was recaptured.

pos_meth	character varying(2)		2 character code for the method of fixing the position. Refer rdb.t_fix_meth_codes.
pos_err	numeric(3,1)		Radius of margin of error of the position (nautical miles)
area	character varying(5)		Area code from rdb.rdb:area_codes where recapture occurred
rec1	character varying(15)		User defined field as defined in the project table.
rec2	character varying(40)		as for rec1
rec3	character varying(10)		as for rec1
rec4	character varying(15)		as for rec1
rec5	character varying(10)		as for rec1
rec6	character varying(40)		as for rec1
fisher_name	character varying(60)		Name of person who caught/returned the tag
fisher_addr	character varying(120)		Address of person who caught/returned the tag
owner_key	integer	No	Refer to meta record of the dataset
comments	text		Text comment(s) on this return.
comments2	text		Text comment(s) on this return.
position	geometry		Position of vessel as gis point type.

```
Indexes:
   "pk_t_return" PRIMARY KEY, btree (return_key)
   "nx_t_return_position" gist ("position")
Check constraints:
   "enforce_dims_position" CHECK (ndims("position") = 2)
   "enforce_geotype_position" CHECK (geometrytype("position") =
    'POINT'::text OR "position" IS NULL)
   "enforce_srid_position" CHECK (srid("position") = 4326)

Foreign-key constraints:
   "fk_t_return_area_codes" FOREIGN KEY (area)
```

```
 REFERENCES rdb.area_codes(code)
"fk_t_return_rdb_t_gon_sys_desc" FOREIGN KEY (stage_meth)
 REFERENCES rdb.t_gon_sys_desc(stage_meth)
"fk_t_return_species_master" FOREIGN KEY (species)
 REFERENCES rdb.species_master(code)
"fk_t_return_t_project" FOREIGN KEY (proj_id)
 REFERENCES tag.t_project(proj_id)
```

5.5 Table 5: t_tag

Comment: Table of information on individual tags used or to be used in tagging program.

Column	Type	Null?	Description
tag_key	integer	No	Primary key to identify a tag.
tbatch_key	integer		Foreign key to refer to a tag batch
tag_type	character varying(8)		Foreign key to refer to a tag type.
tag_no	character varying(16)		Number or code of the tag.
status_code	character varying(16)		Foreign key to refer to tag status.
issue_date	date		Date when the tag is issued.
proj_id	character varying(10)	No	Code to refer to project record.
release_key	integer		Foreign key to refer to the release event in which the tag and fish is released.
return_key	integer		Foreign key to refer to the return event in which the tag and fish is collected.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Text comment(s) on this record.

Indexes:
"pk_t_tag" PRIMARY KEY, btree (tag_key)

Foreign-key constraints:
"fk_t_tag_t_release" FOREIGN KEY (release_key) REFERENCES tag.t_release(release_key)
"fk_t_tag_t_return" FOREIGN KEY (return_key) REFERENCES tag.t_return(return_key)
"fk_t_tag_t_tag_batch" FOREIGN KEY (tbatch_key) REFERENCES tag.t_tag_batch(tbatch_key)
"fk_t_tag_t_tag_type" FOREIGN KEY (tag_type) REFERENCES tag.t_tag_type(tag_type)

5.6 Table 6: t_link

Comment: Table to contain interpretations of associated release and return data.

Column	Type	Null?	Description
link_key	integer	No	Primary key to identify an interpretation record
return_key	integer		Foreign key to refer to a recapture event.
release_key	integer		Foreign key to refer to a release event.
tag_no	character varying(16)		Number or code of the tag, combined with version it should be unique throughout the whole progamme.
version	smallint		Version control over the interpreted data.
dist	numeric(8,3)		Estimated distance traversed between release and recapture positions. t_project record should record units used, typically kilometers or nautical miles.
dir	character varying(3)		The direction the recaptured animal traveled.
rel_seq	smallint		Number to distinguish sequential release number in multiple release-return situations.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Text comment(s) on this record.

Indexes:
"pk_t_link" PRIMARY KEY, btree (link_key)

Foreign-key constraints:
"fk_t_link_t_release" FOREIGN KEY (release_key)
REFERENCES tag.t_release(release_key)

```
"fk_t_link_t_return" FOREIGN KEY (return_key)
 REFERENCES tag.t_return(return_key)
```

5.7 Table 7: t_tag_batch

Comment: Table to contain tag batch information, each order is treated as a batch.

Column	Type	Null?	Description
tbatch_key	integer	No	Primary key to identify a tag batch.
tag_type	character varying(8)		Foreign key to refer to a tag type.
supplier_key	integer		Foreign key to refer to a tag supplier.
proj_id	character varying(10)		Identifier to link to project table.
order_no	character varying(32)		Order no for purchasing the tag batch.
date_received	date		Date when tag batch is received.
min_no	character varying(32)		Minimum tag number within the tag batch.
max_no	character varying(32)		Maximum tag number within the tag batch.
colour	character varying(16)		Tag colour
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Text comment(s) on this record.

Indexes:
"pk_t_tag_batch" PRIMARY KEY, btree (tbatch_key)

Foreign-key constraints:
"fk_t_tag_batch_t_supplier" FOREIGN KEY (supplier_key) REFERENCES tag.t_supplier(supplier_key)
"fk_t_tag_batch_t_tag_type" FOREIGN KEY (tag_type) REFERENCES tag.t_tag_type(tag_type)

5.8 Table 8: t_tag_type

Comment: Table to identify different types of tags used.

Column	Type	Null?	Description
tag_type	character varying(8)	No	Primary key to identify the tag type.
type_desc	text		Descriptive text for the tag type.
legend	character varying(80)		Tag legend.
prefix	character varying(31)		Tag number prefix.
suffix	character varying(31)		Tag number suffix.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Text comment(s) on this record.

Indexes:
"pk_t_tag_type" PRIMARY KEY, btree (tag_type)

5.9 Table 9: t_status

Comment: Table to contain detailed tag status information.

Column	Type	Null?	Description
status_code	character varying(16)	No	Primary key to identify status of tags.
status_desc	character varying(64)		Descriptive text about the status.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Text comment(s) on this record.

Indexes:
"pk_t_status" PRIMARY KEY, btree (status_code)

5.10 Table 10: t_tag_status

Comment: Table to contain detailed tag status information for each release or return.

Column	Type	Null?	Description
tag_key	integer	No	Part of primary key to identify a tag.
status_code	character varying(16)	No	Part of primary key to identify status of tags.
status_date	date		Start date when this status begins.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Any comments regarding to the status of the tag.

Indexes:
"pk_t_tag_status" PRIMARY KEY, btree (tag_key, status_code)

Foreign-key constraints:
"fk_t_tag_status_t_tag" FOREIGN KEY (tag_key)
REFERENCES t_tag(tag_key)
"fk_tag_status" FOREIGN KEY (status_code)
REFERENCES tag.t_status(status_code)

5.11 Table 11: t_supplier

Comment: Table to contain information about tag suppliers.

Column	Type	Null?	Description
supplier_key	integer	No	Primary key to identify a tag supplier.
supplier_name	character varying(64)		Name of the supplier.
supplier_code	character varying(16)		Short name for the supplier.
contact	character varying(64)		Persons to contact.
phone	character varying(32)		Phone number(s).
email	character varying(64)		email address if any.
address	character varying(64)		Postal address.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Text comment(s) on this record.

Indexes:
"pk_t_supplier" PRIMARY KEY, btree (supplier_key)

5.11 Table 11: t_tag_photo

Comment: Table to contain locations of tag photos.

Column	Type	Null?	Description
tphoto_key	integer	No	Primary key to identify a tag photo.
tag_type	character varying(32)		Primary key to identify the tag type.
photo_path	character varying(128)		Directory path where the photo file is located.
photo_name	character varying(32)		The photo file name.
owner_key	integer	No	Refer to meta record of the dataset.

Indexes:
"pk_t_tag_photo" PRIMARY KEY, btree (tphoto_key)

Foreign-key constraints:
"fk_t_tag_photo_t_tag_type" FOREIGN KEY (tag_type)
REFERENCES tag.t_tag_type(tag_type)

5.13 Table 13: t_fish_photo

Comment: Table to contain locations of tagged animal photos.

Column	Type	Null?	Description
fphoto_key	integer	No	Primary key to identify a fish photo.
release_key	integer		Foreign key to refer to a release event.
return_key	integer		Foreign key to refer to a recapture event.
photo_path	character varying(100)		Directory path where the photo file is located.
photo_name	character varying(32)		Photo file name.
owner_key	integer	No	Refer to meta record of the dataset.
comments	character varying(256)		Comments on the animal photo.

Indexes:
"pk_t_fish_photo" PRIMARY KEY, btree (fphoto_key)

Foreign-key constraints:
"fk_t_fish_photo_t_release" FOREIGN KEY (release_key)
REFERENCES tag.t_release(release_key)
"fk_t_fish_photo_t_return" FOREIGN KEY (return_key)
REFERENCES tag.t_return(return_key)

5.14 Table 14: t_lfreq

Comment: Table to store length frequency data collected during a tagging programme.

Column	Type	Null?	Description
lfreq_key	integer	No	Primary key to identify a length frequency record.
proj_id	character varying(10)	No	Foreign key to refer to a tagging programme.
station_code	character varying(12)	No	Identifier to link to releases table.
species	character(3)	No	Valid 3 letter species code.
meas_meth	integer		Code describing the method used to derive the length of the animal.
lgth	numeric(4,1)	No	Length in cm (to 1 mm as decimal if required).
no_m	integer		No of males at this length.
no_f	integer		No of females at this length.
no_t	integer	No	Total of males, females & unsexed animals at this length.
owner_key	integer	No	Refer to meta record of the dataset.

Indexes:
"pk_t_lfreq" PRIMARY KEY, btree (lfreq_key)

Foreign-key constraints:
"fk_t_lfreq_species_master" FOREIGN KEY (species) REFERENCES rdb.species_master(code)
"fk_t_lfreq_t_project" FOREIGN KEY (proj_id) REFERENCES tag.t_project(proj_id)

5.15 Table 15: t_catch

Comment: Table to store catch data from tagging stations.

Column	Type	Null?	Description
catch_key	integer	No	Primary key to identify a catch record.
proj_id	character varying(10)	No	Foreign key to refer to a tagging programme.
station_code	character varying(20)	No	Identifier to link to releases table.
ctch_date	date		Date of catch.
area	character varying(4)		Area code from rdb.area_codes where catch made.
method	character varying(2)		Method code from rdb.meth_codes used to for catch.
species	character(3)	No	Valid 3 letter species code.
weight	numeric(6,1)		Weight in kg.
samp_wt	numeric(6,1)		Weight of sample used for tagging release or return.
tag_link	character varying(8)	No	Flag to mark if catch relates to tag releases or returns.
owner_key	integer	No	Refer to meta record of the dataset.
comments	text		Catch comments.

Indexes:
"pk_t_catch" PRIMARY KEY, btree (catch_key)

Foreign-key constraints:
"fk_t_catch_rdb_area_codes" FOREIGN KEY (area)
REFERENCES rdb.area_codes(code)
"fk_t_catch_rdb_curr_spp" FOREIGN KEY (species)
REFERENCES rdb.curr_spp(code)
"fk_t_catch_t_project" FOREIGN KEY (proj_id)
REFERENCES tag.t_project(proj_id)

View v_release

Comment: View showing release information plus tag details.

Column	Type
release_key	integer
proj_id	character varying(10)
station_code	character varying(12)
trip_code	character varying(9)
vessel	character varying(25)
staff	character varying(80)
min_depth	integer
max_depth	integer
area	character varying(5)
lat	numeric(7,3)
n_s	character varying(1)
lon	numeric(8,3)
e_w	character varying(1)
dlat	numeric(8,6)
dlon	numeric(9,6)
pos_meth	character varying(2)
pos_err	numeric(6,3)
method	character varying(2)
species	character(3)
rel_date	date
rel_time	integer
lgth	numeric(5,1)
lgth_meth	character varying(1)
rel_width	numeric(5,2)
width_meth	character varying(1)

```
weight          numeric(7,3)

sex             character varying(1)

stage_meth      character varying(2)

stage           character varying(2)

age_flag        character varying(1)

no_tags         smallint

condition       character varying(6)

rel1            character varying(20)

rel2            character varying(40)

rel3            character varying(10)

rel4            character varying(10)

rel5            character varying(10)

rel6            character varying(40)

owner_key       integer

comments        text

position        geometry

tag_key         integer

tbatch_key      integer

tag_type        character varying(32)

tag_no          character varying(16)

status_code     character varying(16)

issue_date      date

return_key      integer

View definition:
SELECT r.release_key, r.proj_id, r.station_code, r.trip_code,
        r.vessel, r.staff, r.min_depth, r.max_depth, r.area,
        r.lat, r.n_s, r.lon, r.e_w, r.dlat, r.dlon, r.pos_meth,
        r.pos_err, r.method, r.species, r.rel_date, r.rel_time,
        r.lgth, r.lgth_meth, r.rel_width, r.width_meth, r.weight,
        r.sex, r.stage_meth, r.stage, r.age_flag, r.no_tags,
        r.condition, r.rel1, r.rel2, r.rel3, r.rel4, r.rel5,
        r.rel6, r.owner_key, r.comments, r."position", t.tag_key,
        t.tbatch_key, t.tag_type, t.tag_no, t.status_code,
        t.issue_date, t.return_key
```

```
  FROM tag.t_release r, tag.t_tag t
 WHERE r.release_key = t.release_key;
```

See tables t_release and t_tag for column comments.

View v_return

Comment: View showing return information plus tag details.

Column	Type
return_key	integer
proj_id	character varying(10)
station_code	character varying(12)
trip_code	character varying(9)
vessel	character varying(30)
rec_date	date
rec_date_err	character varying(8)
rec_time	integer
depth	integer
rec_lgth	numeric(5,1)
lgth_meth	character varying(1)
rec_width	numeric(5,2)
width_meth	character varying(1)
species	character(3)
rec_weight	numeric(8,4)
sex	character varying(1)
stage_meth	character varying(2)
stage	character varying(2)
condition	character varying(6)
age_flag	character varying(1)
no_tags	integer
method	character varying(2)
rec_lat	numeric(7,3)
rec_n_s	character varying(1)
rec_lon	numeric(8,3)
rec_e_w	character varying(1)

```
dlat            numeric(8,6)

dlon            numeric(9,6)

pos_meth        character varying(2)

pos_err         numeric(3,1)

area            character varying(5)

rec1            character varying(15)

rec2            character varying(40)

rec3            character varying(10)

rec4            character varying(15)

rec5            character varying(10)

rec6            character varying(40)

fisher_name     character varying(60)

fisher_addr     character varying(120)

owner_key       integer

comments        text

comments2       text

tag_key         integer

tbatch_key      integer

tag_type        character varying(32)

tag_no          character varying(16)

status_code     character varying(16)

issue_date      date

release_key     integer

View definition:
SELECT r.return_key, r.proj_id, r.station_code, r.trip_code,
r.vessel, r.rec_date, r.rec_date_err, r.rec_time, r.depth,
r.rec_lgth, r.lgth_meth, r.rec_width, r.width_meth, r.species,
r.rec_weight, r.sex, r.stage_meth, r.stage, r.condition,
r.age_flag, r.no_tags, r.method, r.rec_lat, r.rec_n_s, r.rec_lon,
r.rec_e_w, r.dlat, r.dlon, r.pos_meth, r.pos_err, r.area, r.rec1,
r.rec2, r.rec3, r.rec4, r.rec5, r.rec6, r.fisher_name,
r.fisher_addr, r.owner_key, r.comments, r.comments2, t.tag_key,
t.tbatch_key, t.tag_type, t.tag_no, t.status_code, t.issue_date,
t.release_key
```

```
  FROM tag.t_return r, tag.t_tag t
 WHERE r.return_key = t.return_key;

See tables t_return and t_tag for column comments
```

6. Tag Business Rules

6.1 Introduction to business rules

The following are a list of business rules pertaining to the **tag** database (see Section 2.2 “Data loading and Validation”). A business rule is a written statement specifying what the information system (i.e., any system that is designed to handle tag data) must do or how it must be structured.

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

Fact rules are shown on the ERD by the cardinality (e.g., one-to-many) of table relationships. Formula and validation rules are implemented by referential constraints, range checks, and algorithms both in the database and during data validation.

Because of the generalised nature of the tag database schema, business rules can not be defined for data in the user-defined fields. For such fields, business rules are often specified in the definition fields in the *t_project* table.

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 preloading software. The use of the word ‘should’ in relation to these validation checks means that a warning message is generated when a value falls outside this range and the data are then checked further in relation to this value.

A data range rule on an attribute only applies when the attribute is not null.

6.2 Summary of rules

t_meta

owner_key	Must be unique within the tag database.
owner_name	Should be a valid name of an organization or person.
project_code	refer to t_project.proj_code.
load_date	Must be a legitimate date

t_project

proj_id	Must be unique within the tag database.
proj_code	Project code should be a valid code within the NIWA and/or MFish project management system.
species	Must be a valid species code as listed in the *species_master* table in the **rdb** database.
date_s	Must be a legitimate date.
date_f	Must be a legitimate date. Multiple column checks on date: The start date should not be later than the finish date.
areas	Each of the listed area codes must be a valid code as listed in the area_codes table in the rdb database.
rel1 - rel5 **rec1 - rec5**	Must be used to describe what values are being stored in the user-defined fields in the t_release and t_return tables respectively. Default value - “not used”.
owner_key	Must be a valid integer value listed in t_meta table.

t_release

release_key	Must be unique within the tag database.
proj_id	Must be a valid proj_id listed in t_project table.
min_depth	Must be a number greater than 0.
max_depth	Must be a number greater than 0. Multiple column checks on depth: The minimum depth should be less than or equal to the maximum depth.
area	Must be a valid area code as listed in the area_code table in the rdb database.
lat	Must be a valid latitude ranging from 0 to 90.
n_s	Must be equal to either an “N” or a “S” if not null, and should not be Null if lat is not null.
lon	Must be a valid longitude ranging from 0 to 180.
e_w	Must be equal to either an “E” or a “W” if not null, and should not be Null if lon is not null.
dlat	Must be a valid latitude ranging from 90 to -90 degrees.

dlon	Must be a valid longitude ranging from 0 to 360 degrees. **Multiple column checks on lat, n_s, lon, E_W, dlat, dlon, area: The latitude and longitude should be within the area recorded.**
method	Must be a valid gear method code as listed in the meth_codes table in the rdb database.
species	Must be a valid species code as listed in the species_master table in the rdb database.
rel_date	Must be a valid date.
rel_time	Must be a valid 24 hour time ranging 0 to 2359.
lgth	Must be a number greater than 0, and should be less than the maximum recorded length for the species as recorded in the curr_spp table in the rdb database.
lgth_meth	Must be a valid fish measurement code as listed in the t_fish_meas_codes table in the rdb database.
rel_width	Must be a number greater than 0.
width_meth	Must be a valid fish measurement code as listed in the t_fish_meas_codes table in the rdb database.
weight	Must be a number greater than 0.
sex	Sex code. Must be a valid sex code as listed in the t_sex_codes table in the rdb database.
stage_meth	Must be a valid gonad staging system as listed in the t_gon_sys_desc table in the rdb database.
stage	Gonad (or life cycle) stage. Must be a valid code as listed in the t_gon_stg_meth table in the rdb database.
rel1 - rel5	User defined fields. Descriptions of the fields usage must be listed in the t_project table.
owner_key	Must be a valid integer value listed in t_meta table.

t_return

return_key	Must be unique within the tag database.
proj_id	Must be a valid proj_id listed in t_project table.
rec_date	Must be a valid date and should be on or after the initial release of the tagged animal.
rec_time	Must be a valid 24 hour time ranging 0 to 2359.
depth	Must be a number greater than 0.
rec_lgth	Must be a number greater than 0, and should be less than the maximum recorded length for the species as recorded in the curr_spp table in the rdb database.
lgth_meth	Must be a valid fish measurement code as listed in the t_fish_meas_codes table in the rdb database.
rec_width	Must be a number greater than 0.
width_meth	Must be a valid fish measurement code as listed in the t_fish_meas_codes table in the rdb database.
rec_weight	Must be a number greater than 0.
sex	Must be a valid sex code as listed in the t_sex_codes table in the rdb database.

stage_meth	Must be a valid gonad staging system as listed in the t_gon_sys_desc table in the rdb database.
stage	Gonad (or life cycle) stage. Must be a valid code as listed in the t_gon_stg_meth table in the rdb database.
method	Must be a valid gear method code as listed in the meth_codes table in the rdb database.
rec_lat	Must be a valid latitude ranging from 0 to 90.
rec_n_s	Must be equal to either “N” or “S”.
rec_lon	Must be a valid longitude ranging from 0 to 180.
rec_e_w	Must be equal to either an “E” or a “W”.
dlat	Must be a valid latitude ranging from 90 to -90 degrees.
dlon	Must be a valid longitude ranging from 0 to 360 degrees.
area	Must be a valid area code as listed in the *area_codes* table in the **rdb** database. **Multiple column checks on rec_lat, N_S, rec_lon, E_W, dlat, dlon, area:** **The latitude and longitude should be within the area recorded.**
pos_meth	Must be a valid position fixing method code as listed the t_fix_meth_codes table of the rdb database.
rec1 - rec5	User defined fields. Descriptions of the fields usage must be listed in the *t_project* table.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_tag

tag_key	Must be unique within the tag database.
tbatch_key	Must be a valid integer value listed in *t_tag_batch* table.
tag_type	Must be a valid type code listed in *t_tag_type* table.
status_code	Must be a valid status code listed in *t_status* table.
issue_date	Must be a valid date.
proj_id	Must be a valid proj_id listed in *t_project* table.
release_key	Must be a valid integer value listed in *t_release* table.
return_key	Must be a valid integer value listed in *t_return* table.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_link

link_key	Must be unique within the tag database.
return_key	Must be a valid integer value listed in *t_return* table.
release_key	Must be a valid integer value listed in *t_release* table.
dist	Distance must be a number greater than 0.
dir	Should be a valid compass direction involving the characters “N”, “E”, “S”, and “W”; e.g. “ENE”, or an integer representing a valid direction in degrees from 0 to 359.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_tag_batch

tbatch_key	Must be unique within the tag database.
tag_type	Must be a valid type code listed in *t_tag_type* table.
supplier_key	Must be a valid integer value listed in *t_supplier* table.
proj_id	Must be a valid proj_id listed in *t_project* table.
date_received	Must be a legitimate date.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_tag_type

tag_type	Must be unique within the tag database.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_status

status_code	Must be unique within the tag database.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_tag_status

tag_key	Must be a valid integer value listed in *t_tag* table.
status_code	Must be a valid status code listed in *t_status* table, the combination of tag_key and status_code must be unique.
status_date	Must be a legitimate date.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_supplier

supplier_key	Must be unique within the tag database.
email	Should contain a valid email address.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_tag_photo

tphoto_key	Must be unique within the tag database.
tag_type	Must be a valid type code listed in *t_tag_type* table.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_fish_photo

fphoto_key	Must be unique within the tag database.
release_key	Must be a valid integer value listed in *t_release* table.
return_key	Must be a valid integer value listed in *t_return* table.

owner_key	Must be a valid integer value listed in *t_meta* table.

t_lfreq

lfreq_key	Must be unique within the tag database.
proj_id	Must be a valid proj_id listed in *t_project* table.
station_code	Should be a station code that has been used in either the *t_release* or the *t_return* table.
species	Must be a valid species code as listed in the *species_master* table in the **rdb** database.
meas_meth	Must be a valid fish (animal) measurement method code as listed in the *meas_meth* table in the **rdb** database.
lgth	Must be a number greater than 0.
no_m	Must be a number greater than or equal to 0.
no_f	Must be a number greater than or equal to 0.
no_t	Must be a number greater than 0. **Multiple column check on no_m, no_f and no_t.** The number in *no_t* should be equal to or greater than the sum of *no_m* and *no_f*.
owner_key	Must be a valid integer value listed in *t_meta* table.

t_catch

catch_key	Must be unique within the tag database.
proj_id	Must be a valid proj_id listed in *t_project* table.
station_code	Should be a station code that has been used in either the *t_release* or the *t_return* table.
species	Must be a valid species code as listed in the *species_master* table in the **rdb** database.
weight	Must be a number greater than 0.
samp_wt	Must be a number greater than 0, and less than or equal to *weight*.
tag_link	Must be equal to either “release” or “return”.
owner_key	Must be a valid integer value listed in *t_meta* table.

7 Acknowledgements

The authors would like to thank Dave Banks for his editorial contribution to the original document.

8 References

Crossland, J. 1982: Tagging of marine fishes in New Zealand. *Fisheries Research Division Occasional Publication No 33*. 19p

Murray, T. 1990: Fish-marking techniques in New Zealand. *American Fisheries Symposium 7*:737--745

Wood, B. A. 1993: Marine Research database documentation. 10. tag. *MAF Fisheries Greta Point Internal Report No. 216* 13p.

Appendix 1 – Reference code tables

Codes for attributes lgth_meth and width_meth from rdb:t_fish_meas_codes

fish_meas_code	description
1	Fork Length
2	Total Length
3	Standard Length
4	Mantle Length (squid)
5	Pelvic Length (rays)
6	Carapace Width
7	Shell Height
8	Shell Length
B	Carapace Length - Orbit to Carapace notch (scampi)
G	Tip of snout to posterior end of dorsal fin (Ghost sharks)
E	Eye to Fork Length (billfish)
J	Lower Jaw to Fork Length (billfish)
O	Orb Length - length across the eye (billfish)
C	Carapace Length - Base of antennal platform to posterior margin
W	Tail Width - as legally defined for red rock lobsters
L	Tail Length - as legally defined for red rock lobsters

stage_meth codes and associated *stage* codes from rdb: t_gon_stg_meth

stage_meth	stage	description
CF	MM	Males
CF	BF	Berried female
CF	IF	Immature female
CF	MF	Mature female, setae greater than 6mm
CF	SC	Scattered - spent female
CF	UF	Unidentified stage female
RL	0	Hermaphrodite or indeterminate
RL	1	Male
RL	2	Immature female
RL	3	Mature female, setae greater then 6mm
RL	4	Berried female, no eyes on berry
RL	5	Berried female, eyes in berry visible
RL	6	Spent female, with infertile/unhatched eggs visible
RL	7	Spent female, no eggs visible
RL	9	Female, maturity not determined

Codes for attribute *pos_meth* from rdb:t_fix_meth_codes

fix_meth_code	description
01	Radar
02	Dead reckoning
03	Astrofix
04	Transect marks
05	Radio (RDF)
06	Radar and RDF
07	SatNav
08	Global Positioning Satelite (GPS)
09	Local knowledge
10	GPX