Database documentation: aer_sight

David Fisher & Paul Taylor

NIWA Fisheries Data Management
Database Documentation Series
Updated – 20 May 2015

Revision History

Version	Change	Date	Responsible
1.0	Release	1995	Paul Taylor
2.0	Updated document to standard format, added table t_set, added business rules.	7-Sep-2001	David Fisher
2.1	Added new codes for t_set.rst 8-12. Changed data types: t_set est_ton_pilot and land_ton from integer, land_sp from smallint. t_species_code from smallint.	18-Jan-2012	David Fisher for Paul Taylor
2.1	Minor spelling corrections to column comments in section 5.	17-Jan-2014	David Fisher
3.1	Postgres version	May 2015	D Fisher & F Wei

Table of Contents

1 Introduction to the Database Document 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 is a brief introduction to the aerial sightings database **aer_sight**, and is part of the database documentation series produced by NIWA. It supersedes the previous documentation by Taylor (1995)[1] 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 link together and their relationship with other databases.

This document is intended as a guide for users and administrators of the **aer_sight** database, and should be used in conjunction with the technical report (Taylor *in preparation)* for the most comprehensive background on the aerial sightings database and its uses.
This database has been implemented as a schema within the Postgres database called **fish**.

Access to this database is restricted to specific nominated personnel as specified in the current Data Management contract between MPI and NIWA. Any requests for data should in the first instance be directed to MPI (email: rdm@mpi.govt.nz).

2 The Aerial Sightings Database

2.1 Data sources

The **aer_sight** database contains records of search effort and sightings of pelagic schooling species (mainly skipjack tuna, kahawai, blue mackerel, jack mackerel and trevally) from pilots spotting fish schools for purse-seiners around New Zealand. Aerial sightings data are not collected according to experimental design. They are captured opportunistically by pilots of light aircraft employed as fish spotters in purse-seining operations. The majority of effort is centred in the Bay of Plenty (BOP), Golden and Tasman Bays, the Kaikoura coast and to a lesser extent, the South Taranaki coast. The time series of these data begins in June 1976, and is ongoing.

[1] Taylor, P., 1995: Database documentation: 13. aerial sightings. *NIWA Internal Report No. 242.* 25p.

2.2 Data validation

While the **aer_sight** database enforces data validation and integrity rules with the use of referential constraints and range checks, data go though rigorous data validation and error checking process before being entered.

This process includes instructions for data recording, simple data validation using the **checkq**[2] validation program language and C programming language scripts. See Appendix 1 for a more detailed description of the processes involved.

2.3 Uses of the aerial sightings database

Aerial sightings data can be used for a range of outputs, from profiles of pelagic species distribution and purse-seine activity to providing background information on stock assessments of these species (Taylor *in preparation).* Probably the most important output is time series of relative abundance which can indicate fluctuations in their abundance from year to year (Bradford & Taylor 1995). Commercial and recreational interests request various types of data reports several times a year.

3 Data Structures

3.1 Table relationships

The ERD for **aer_sight** (Figure 1) shows the logical structure[3] of the database and it's entities (each entity is implemented as a database *table)* and relationships between these tables and tables in other databases. This schema is valid regardless of the database system chosen, and it can remain correct even if the Database Management System (DBMS) is changed. The ERDs in this document show attributes within tables with generic data-types. Each table represents an object, event, or concept in the real world that is selected to be 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[4].

The **aer_sight** database is implemented as a relational database; i.e., each table is a special case of the mathematical construct known as a *relation* and hence elementary relation theory is used to deal with the data within tables and the relationships between them. There are three types of relationships possible between tables, but only two exist in **aer_sight**: one-to-many[5], and one to one. These relationships can be seen in ERDs by connecting a single line (indicating 'many') from the child table; e.g., *t_flight*, to the parent table; e.g., *t_flight_group,* with an arrowhead (indicating 'one') pointing to the parent.

---

[2]See local Unix manual page on **checkq**

[3] Also known as a database *schema.*

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

[5] 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 landing in *t_flight_group* can have many catches in *t_flight* but one catch can only come from one landing.

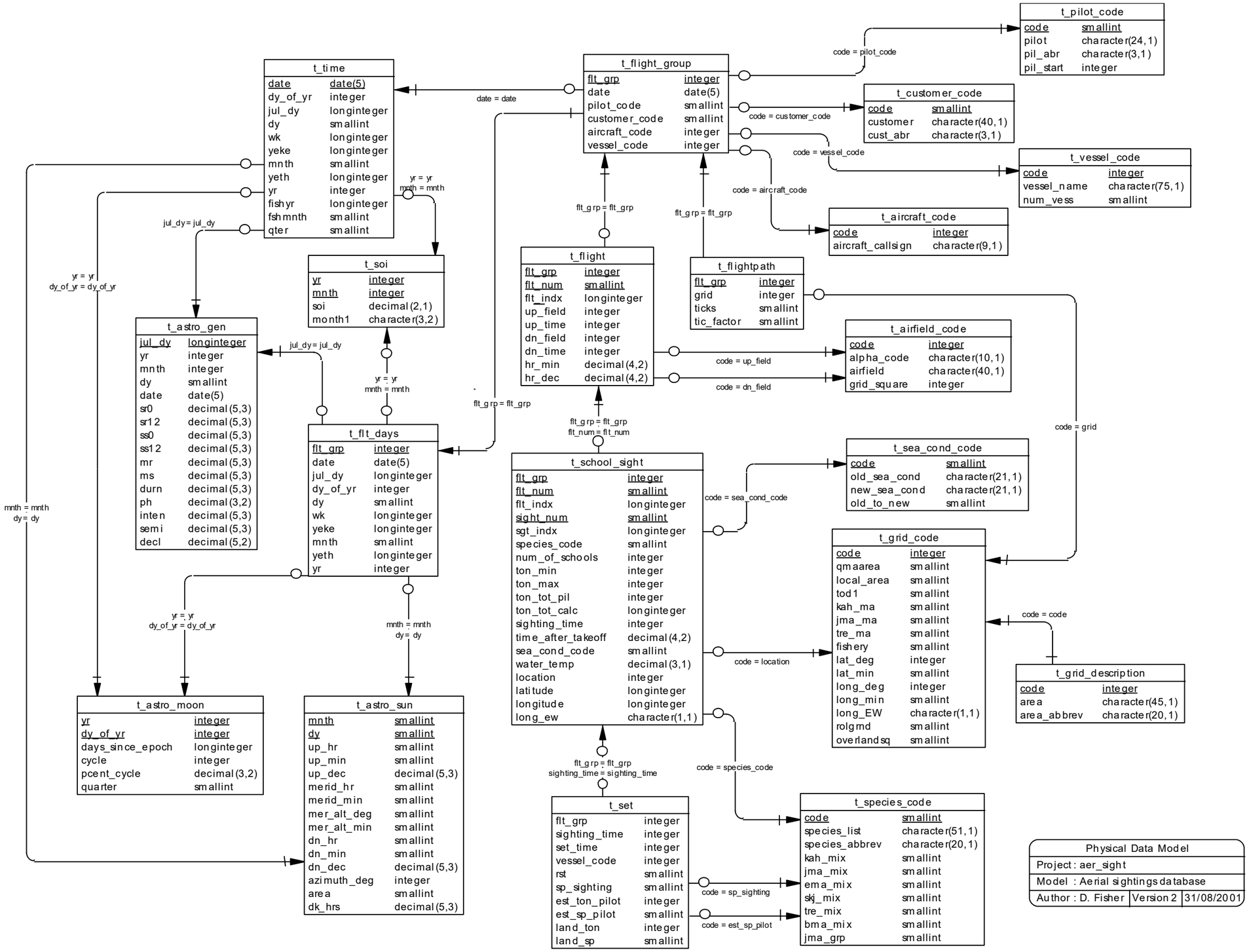

Figure 1: Entity Relationship Diagram (ERD) for the **aer_sight** database

Only one to many relationships are generally acceptable in a normalised database but, in the present case, one to one linkages are also used *(see* Figure 1). There are two main reasons why; to increase accessibility; and to avoid large unwieldy tables. A number of the tables which are linked by one to one relationships can be regarded as ancillary tables, providing either specialised information which is used infrequently (e.g., *t_astro_moon*), or subsets of time series groupings which cannot be readily obtained using standard SQL (e.g., *t_flt_days*). The five main tables contain data collected by the pilots and it is convenient to restrict them accordingly - grouping ancillary data into tables dedicated to specific tasks or selected data precludes development of huge, unwieldy tables, making the available information more transparent to the user. Table comments in Section 5 closely define the uses of all the database tables.

Every relationship has a mandatory or optional aspect to it. 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_flight* and it's child table *t_school_sight.* The symbol 'o' by the child *t_school_sight* means that a flight record can have zero or many school sighting records, while the bar by the parent *t_flight* means that for every school sighting record there must be a matching flight record.

These links are enforced by foreign key constraints[6]. 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 the relationship is lost; or a child record is entered without a parent record

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])

Note that the typographical convention for the above format is that square brackets [] may contain more than one item or none at all. Items stacked between vertical lines | | are options of which one must be chosen.

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

Foreign-key constraints:
"fk_t_flight_t_flight_group" FOREIGN KEY (flt_grp) REFERENCES
t_flight_group(flt_grp)

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

[6] Also known as integrity checks.

Section 5 lists all the **aer_sight** tables as implemented by the Postgres DBMS. As can be seen in the listing of the tables, a table's primary key has a unique index on it. Primary keys are generally listed using the following format:

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

where attribute(s) make up the primary key and the index name is the primary key name. These prevent records with duplicate keys from being inserted into the tables; e.g., a record with an existing *flt_grp* number.

The database listing (Section 5) show that the tables also have indices on many attributes. 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 format:

Indices: index_name btree (*attribute*)

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

Due to difficulties in the collection of these data which are recorded on a voluntary basis by pilots, some historic data will fail referential constraints, and hence some of the expected foreign key constraints cannot be implemented. This is because it has not been possible to associate a school sighting with a particular flight, hence these records were entered into *t_school_sight* with *flt_num* equal to zero. Some sets do not have a set time recorded, and therefore cannot be linked back to a sighting in the table *t_school_sight*.

3.2 Database design

Tables can be summarised under three categories: the main tables, which are five in number and contain the main body of the data; the code tables which enable decoding of the numeric codes used in the main tables - there are nine of these, identified by the suffix "code"; and accessory tables which either allow data extracts to follow particular time frames or contain environmental data used in some analyses - there are six of these.

4 Table Summaries

The tables can be summarised into the following four categories:

4.1 Main tables

The following tables contain the information provided by the fish spotting pilots and the companies they are employed by:

1. t_flight_group - reference information for a group of flights.

2. t_flight - flight duration data and information on takeoff and landing airfields.

3. t_school_sight - data on the fish spotted, including time, location and estimates of the amount.

4. t_set - data on the sets made on schools by fishing vessels.

5. t_flightpath - information on the areas flown and the amount of time elapsed in each.

4.2 Code tables

The following tables enable decoding of the numeric codes used in the main tables, allows grouping of the data in useful ways, and provides textual descriptions of some of the encoded data:

1. t_pilot_code - decoding data for the pilot codes used in *t_flight_group*.

2. t_customer_code - decoding data for the customer codes used in *t_flight_group*.

3. t_aircraft_code - decoding data for the aircraft codes used in *t_flight_group*.

4. t_vessel_code - decoding data for the vessel codes used in *t_flight_group*.

5. t_airfield_code - decoding data for the takeoff and landing airfield codes used in *t_flight*.

6. t_species_code - decoding data for the species codes used in *t_school_sight* and a system for selecting mixed schools of the 5 main species.

7. t_sea_cond_code - decoding data for the sea condition codes used in *t_school_sight* and a grouping code enabling selection of pre-11/85 data according to the new codes.

8. t_grid_code - various sets of grouping codes enabling the selection of data by particular area formats; e.g., QMA, KAH FMA etc.

9. t_grid_description - descriptions of the location and grid codes used in *t_school_sight* and *t_flightpath*.

4.3 Time tables

The following tables provide methods of grouping data by time:

1.t_flt_days - enables selection of data according to various time series (e.g., day of the year, Julian day etc.).

2. t_time - similar data to those contained in *t_flt_days* but includes all days since the beginning of the database through 12/95 - this allows reference to days where no flying was done (useful in some estimates of search effort) and enables update of *t_flt_days*.

4.4 Environmental tables

The following tables provide useful environmental data:

1. t_soi - a monthly time series of southern oscillation indices for the period of the database, for use in analyses requiring environmental variables.

2. t_astro_gen - times of sun- and moon-rise and various other astronomical data.

3. t_astro_moon - moon cycle data.

4. t_astro_sun - sun cycle data.

5 Aerial Sightings Tables

The following is a comprehensive listing, including attribute names, range checks, referentials, indices, and number of records, of all tables in the aerial sightings database.

5.1 Main Tables

5.1.1 Table: t_flight_group

```
Comment: 1st of the five main tables - contains reference data for a
         group of flights.

Column          Type                Null?  Description

flt_grp         smallint              No   Link between the five main
                                           tables and with t_flt_days.

date            date                       Date of the group of
                                           flights.

pilot_code      smallint              No   Numeric code of pilot (&
                                           observers) recording the
                                           data.

customer_code   smallint                   Numeric code for customer
                                           requesting the flight.

aircraft_code   smallint              No   Numeric code for aircraft
                                           being flown.

vessel_code     smallint                   Numeric code for vessel(s)
                                           being assisted.


Indexes:
   "pk_t_flight_group" PRIMARY KEY, btree (flt_grp)
   "nx_t_flight_group_aircraft_code" btree (aircraft_code)
   "nx_t_flight_group_customer_code" btree (customer_code)
   "nx_t_flight_group_date" btree (date)
   "nx_t_flight_group_pilot_code" btree (pilot_code)
   "nx_t_flight_group_vessel_code" btree (vessel_code)
Check constraints:
   "t_flight_group_customer_code_check" CHECK (customer_code IS NULL OR
     customer_code >= 0 AND customer_code <= 99)
   "t_flight_group_date_check" CHECK (date::text > 19760621::text)
   "t_flight_group_flt_grp_check" CHECK (flt_grp > 0)
   "t_flight_group_pilot_code_check" CHECK (pilot_code >= 0 AND
                                            pilot_code <= 99)

Foreign-key constraints:
   "fk_t_flight_group_1" FOREIGN KEY (pilot_code)
    REFERENCES aer_sight.t_pilot_code(code)
   "fk_t_flight_group_2" FOREIGN KEY (aircraft_code)
```

```
 REFERENCES aer_sight.t_aircraft_code(code)
"fk_t_flight_group_3" FOREIGN KEY (customer_code)
 REFERENCES aer_sight.t_customer_code(code)
"fk_t_flight_group_4" FOREIGN KEY (vessel_code)
 REFERENCES aer_sight.t_vessel_code(code)
```

5.1.2 Table 2: t_flight

Comment: 2nd of the five main tables - contains flight duration and airfield data for individual flights - "flt_grp" identifies a group of flights by a pilot on a single day - 10 flights is maximum and usually there are less than five.

Column	Type	Null?	Description
flt_grp	smallint	No	Link between the five main tables & with t_flt_days.
flt_num	smallint	No	Chronological numbering of flights within a group.
flt_indx	integer		Concatenation of flt_grp & flt_num (eg. flt_grp = 1, flt_num = 1, flt_indx = 11) used for simple link with t_school_sight (t_flight.flt_index = t_school_sight.flt_index).
up_field	smallint	No	Airfield used at takeoff.
up_time	smallint	No	Time of takeoff.
dn_field	smallint	No	Airfield used at landing.
dn_time	smallint	No	Time of landing.
hr_min	numeric(4,2)		Duration of flight in hours & minutes ("dn_time" minus "up_time").
hr_dec	numeric(4,2)		Duration of flight in decimal hours ("dn_time" minus "up_time").

Indexes:
- "pk_t_flight" PRIMARY KEY, btree (flt_grp, flt_num)
- "nx_t_flight_dn_field" btree (dn_field)
- "nx_t_flight_dn_time" btree (dn_time)
- "nx_t_flight_flt_indx" btree (flt_indx)
- "nx_t_flight_up_field" btree (up_field)
- "nx_t_flight_up_time" btree (up_time)

Foreign-key constraints:
- "fk_t_flight_t_airfield_code_1" FOREIGN KEY (up_field) REFERENCES aer_sight.t_airfield_code(code)
- "fk_t_flight_t_airfield_code_2" FOREIGN KEY (dn_field) REFERENCES aer_sight.t_airfield_code(code)
- "fk_t_flight_t_flight_group" FOREIGN KEY (flt_grp) REFERENCES aer_sight.t_flight_group(flt_grp)

5.1.3 Table 3: t_school_sight

Comment: 3rd of the five main tables - contains data on the species sighted, estimates of tonnage, location & time of the sighting, and some environmental data.

Column	Type	Null?	Description
flt_grp	smallint	No	Link between the five main tables & with t_flt_days.
flt_num	smallint	No	Refers to flt_num in t_flight (i.e. chronological numbering of flights within a group). Equals 0 when sighting time is null or is outside the range of flight times or no sightings were made.
flt_indx	integer		Concatenation of flt_grp & flt_num (eg. flt_grp = 1, flt_num = 1, flt_indx = 11) used for simple link with t_flight (t_school_sight.flt_index = t_flight.flt_index).
sight_num	smallint	No	Chronological numbering of sightings during a flight. Equals 0 where no sightings were made.
sgt_indx	integer		Concatenation of flt_grp, flt_indx, & sight_num.
species_code	smallint		Numeric code of sighted species - decoded using t_species_code.
num_of_schools	smallint		The number of schools observed in the sighting.
ton_min	smallint		The minimum of the tonnage range of the schools observed.
ton_max	smallint		The maximum of the tonnage range of the schools observed.
ton_tot_pil	smallint		The estimate of the total tonnage present in the sighting determined by the pilot.

ton_tot_calc	integer	The estimate of the total tonnage calculated using the mean of ton_min & ton_max times the num_of_schools.
sighting_time	smallint	The time that the sighting is made - time is recorded as NZDT during periods that it is effective - no adjustment is made to standardise to NZST.
time_after_takeoff	numeric(4,2)	Time in decimal hours after take off that sighting was made.
sea_cond_code	smallint	Numeric code of sea condition - 1 is calm/slight, 2 is moderate & 3 is rough - decoded using t_sea_cond_code.
water_temp	numeric(3,1)	Sea surface temperature (degrees C).
location	smallint	Numeric code of half degree square location of the sighting - can be referenced according to various statistical & management areas using t_grid_code or decoded (relative to landmarks) using t_grid_description.
latitude	integer	Latitude (south) of sighting location in degrees and minutes to 2 implied decimal places.
longitude	integer	Longitude of sighting location in degrees and minutes to 2 implied decimal places.
long_ew	character varying(1)	E or W for longitude.
dlat	numeric(8,5)	Latitude of sighting location in decimal degrees.
dlon	numeric(8,5)	Longitude of sighting location in decimal degrees.
position	geometry	Position of sighting location as gis point type.

```
Indexes:
    "pk_t_school_sight" PRIMARY KEY, btree (flt_grp, flt_num, sight_num)
    "nx_t_school_sight_flt_indx" btree (flt_indx)
    "nx_t_school_sight_location" btree ("location")
    "nx_t_school_sight_num_of_school" btree (num_of_schools)
    "nx_t_school_sight_sgt_indx" btree (sgt_indx)
    "nx_t_school_sight_sighting_time" btree (sighting_time)
    "nx_t_school_sight_species_code" btree (species_code)
    "nx_t_school_sight_time_after_ta" btree (time_after_takeoff)
    "nx_t_school_sight_ton_max" btree (ton_max)
    "nx_t_school_sight_ton_min" btree (ton_min)
    "nx_t_school_sight_ton_tot_calc" btree (ton_tot_calc)
    "nx_t_school_sight_ton_tot_pil" btree (ton_tot_pil)
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)
    "t_school_sight_latitude_check" CHECK (latitude IS NULL OR
      latitude::text ~ '[3-4][0-9][0-5][0-9][0-9][0-9]'::text)
    "t_school_sight_longitude_check" CHECK (longitude IS NULL OR
      longitude::text ~ '1[7-8][0-9][0-5][0-9][0-9][0-9]'::text)

Foreign-key constraints:
    "fk_t_school_sight_t_grid_code" FOREIGN KEY ("location")
     REFERENCES aer_sight.t_grid_code(code)
    "fk_t_school_sight_t_species_code" FOREIGN KEY (species_code)
     REFERENCES aer_sight.t_species_code(code)
```

5.1.4 Table 4: t_set

Comment: 4th of the five main tables - contains data on the sets made on schools by fishing vessels.

Column	Type	Null?	Description
flt_grp	smallint	No	Link between the five main tables & with t_flt_days.
sighting_time	smallint		Time sighting was made; = t_school_sight.sighting_time .
set_time	smallint		Time set was made.
vessel_code	smallint		Code for the vessel making the set.
rst	smallint		Code defining the outcome of the shot: 1 = caught(whole school), 2 = some lost some saved, 3 = for skunked, 4 = unknown (pilot left area before shot completed), 5 = caught unknown amount (unavailable from vessel), 6 = let go, 7 = burst net, 8 = ripped net, 9 = caught enough to fill, 10 = spooked, 11 = set made using sonar, implies no assistance from the plane, 12 = broke purse cable.
sp_sighting	smallint		Species estimated by pilot at first sighting.
est_ton_pilot	numeric(4,1)		Tonnage estimated by the pilot. Values since June 1998 are integer, before June 1998 are decimal(4,1).
est_sp_pilot	smallint		Species estimated by the pilot.
land_ton	numeric(4,1)		Tonnage estimated by the vessel after fish brailed. Values since June 1998 are integer, before June 1998 are decimal(4,1).
land_sp	smallint		Species estimated by vessel after fish brailed.

Indexes:

```
   "nx_t_set_flt_grp" btree (flt_grp)
Check constraints:
   "t_set_rst_check" CHECK (rst IS NULL OR rst >= 1 AND rst <= 12)
   "t_set_set_time_check" CHECK (set_time IS NULL OR set_time < 2400)
   "t_set_sighting_time_check" CHECK (sighting_time IS NULL OR
    sighting_time < 2400)

Foreign-key constraints:
   "fk_t_set_t_flight_group" FOREIGN KEY (flt_grp)
    REFERENCES aer_sight.t_flight_group(flt_grp)
   "fk_t_set_t_species_code_1" FOREIGN KEY (sp_sighting)
    REFERENCES aer_sight.t_species_code(code)
   "fk_t_set_t_species_code_2" FOREIGN KEY (est_sp_pilot)
    REFERENCES aer_sight.t_species_code(code)
   "fk_t_set_t_species_code_3" FOREIGN KEY (land_sp)
    REFERENCES aer_sight.t_species_code(code)
```

5.1.5 Table 5: t_flightpath

Comment: 5th of the five main tables - contains records of the half degree squares flown during a group of flights (see t_flight) and the 10-15 minute periods spent therein.

Column	Type	Null?	Description
flt_grp	smallint	No	Link between the five main tables & with t_flt_days.
grid	smallint	No	Numeric code of half degree square location of the sighting - can be referenced according to various statistical & management areas using t_grid_code or decoded (relative to landmarks) using t_grid_description.
ticks	smallint		Number of 10-15 minute periods spent within a half degree square.
tic_factor	smallint		Similar to "ticks" but NULLs in "ticks" replaced with a value of "1" for easy addition - "ticks" were not recorded before 1/11/85, only entries into each grid square during a flight; therefore addition of "1"s gives approximation to "ticks" though less accurate.

Indexes:
   "nx_t_flightpath_flt_grp" btree (flt_grp)
   "nx_t_flightpath_grid" btree (grid)
   "nx_t_flightpath_tic_factor" btree (tic_factor)

Foreign-key constraints:
   "fk_t_flightpath_t_flight_group" FOREIGN KEY (flt_grp)
    REFERENCES aer_sight.t_flight_group(flt_grp)
   "fk_t_flightpath_t_grid_code" FOREIGN KEY (grid)
    REFERENCES aer_sight.t_grid_code(code)

5.2 Code Tables

5.2.1 Table: t_pilot_code

Comment: Reference data for pilot codes.

Column	Type	Null?	Description
code	smallint	No	"pilot_code" in t_flight_group.
pilot	character varying(24)	No	Pilot name.
pil_abr	character varying(3)	No	Abbreviated pilot name.
pil_start	smallint		Number of days after 1/1/76 that pilot started spotting.

Indexes:
   "pk_t_pilot_code" PRIMARY KEY, btree (code)
Check constraints:
   "t_pilot_code_code_check" CHECK (code >= 0 AND code <= 99)

5.2.2 Table: t_customer_code

```
Comment: Reference data for customers of pilots.

Column          Type                 Null?  Description

code            smallint               No   "customer_code" in
                                            t_flight_group.

customer        character varying(40)  No   Customer name.

cust_abr        character varying(3)   No   Abbreviated customer name.


Indexes:
   "pk_t_customer_code" PRIMARY KEY, btree (code)
   "ui_t_customer_code_cust_abr" UNIQUE, btree (cust_abr)
   "ui_t_customer_code_customer" UNIQUE, btree (customer)
Check constraints:
   "t_customer_code_code_check" CHECK (code >= 0 AND code <= 99)
```

5.2.3 Table: t_aircraft_code

Comment: Reference data for aircraft.

Column	Type	Null?	Description
code	smallint	No	"aircraft_code" in t_flight_group.
aircraft_callsign	character varying(9)	No	Registered callsign of from 1 to 3 aircraft concatenated together.

Indexes:
"pk_t_aircraft_code" PRIMARY KEY, btree (code)
"ui_t_aircraft_code_aircraft_callsign" UNIQUE, btree (aircraft_callsign)

5.2.4 Table: t_vessel_code

Comment: Reference data for vessel codes.

Column	Type	Null?	Description
code	smallint	No	"vessel_code" in t_flight_group.
vessel_name	character varying(75)	No	Name of vessel aggregate.
num_vess	smallint		Number of vessels in aggregate.

Indexes:
"pk_t_vessel_code" UNIQUE, btree (code)

5.2.5 Table: t_airfield_code

Comment: Reference data for airfields and landing strips.

Column	Type	Null?	Description
code	smallint	No	"up_field" and "dn_field" in t_flight.
alpha_code	character varying(10)		Civil Aviation Authority & spotter-pilot-created codes of airfields & strips.
airfield	character varying(40)		Name of airfield or strip.
grid_square	smallint		Half degree grid square location of airfield or strip.

Indexes:
"pk_t_airfield_code" PRIMARY KEY, btree (code)
"nx_t_airfield_code_airfield" btree (airfield)
"nx_t_airfield_code_alpha_code" btree (alpha_code)
"nx_t_airfield_code_grid_square" btree (grid_square)

5.2.6 Table: t_species_code

Comment: Reference data for fish species. Note these are not the research 3 letter species codes.

Column	Type	Null?	Description
code	smallint	No	"species_code" in t_school_sight.
species_list	character varying(51)		Species names.
species_abbrev	character varying(20)	No	Abbreviated species names.
kah_mix	smallint		Flag to group schools of kahawai mixed with any other species (1 = TRUE).
jma_mix	smallint		Flag to group schools of one to three species of jack mackerel mixed with any species other than jack mackerels (1 = True).
ema_mix	smallint		Flag to group schools of blue mackerel mixed with any other species (1 = True).
skj_mix	smallint		Flag to group schools of skipjack tuna mixed with any other species (1 = TRUE).
tre_mix	smallint		Flag to group schools of trevally mixed with any other species (1 = TRUE).
bma_mix	smallint		Flag to group schools of blue maomao mixed with any other species (1 = TRUE).
jma_grp	smallint		Flag to group various jack mackerel species, but not in schools mixed with any other species (1 = True).

Indexes:
"pk_t_species_code" UNIQUE, btree (code)

5.2.7 Table: t_sea_cond_code

Comment: Reference data for sea condition including means of standardising old system (pre- 1/11/85) with the new.

Column	Type	Null?	Description
code	smallint	No	"sea_cond_code" in t_school_sight.
old_sea_cond	character varying(21)	No	Codes used in old system.
new_sea_cond	character varying(21)		Codes used in new system.
old_to_new	smallint		Means of standardising old system with the new.

Indexes:
   "pk_t_sea_cond_code" PRIMARY KEY, btree (code)
Check constraints:
   "t_sea_cond_code_code_check" CHECK (code >= 1 AND code <= 4)

5.2.8 Table: t_grid_code

Comment: Reference data for the half degree grid used for locating sightings and sightings effort.

Column	Type	Null?	Description
code	smallint	No	"location" in t_school_sight and "grid" in t_flightpath.
qmaarea	smallint		Grid squares indexed according to the FMAs.
local_area	smallint		Grid squares indexed as follows: 1 = Nth Cape to Great Barrier, 2 = BOP , 3 = Rolling Ground/South Taranaki Bight, 4 = Kahurangi-Golden & Tasman Bays, 5 = Brothers to Kekerengu, 6 = Kaikoura.
tod1	smallint		System to sort sightings between 36.30S and 38S as follows: 1 = West Coast, 2 = Hauraki Gulf, 3 = East Coast within FMA1 and 4 = FMA2.
kah_ma	smallint		Grid squares indexed according to the kahawai fishstocks.
jma_ma	smallint		Grid squares indexed according to the jack mackerel fishstocks.
tre_ma	smallint		Grid squares indexed according to the trevally fishstocks.
fishery	smallint		Northern fishery = 1, southern fishery = 2.
lat_deg	smallint		Degrees of latitude at the grid square.
lat_min	smallint		Minutes of latitude at the grid square.
long_deg	smallint		Degrees of longitude at the grid square.
long_min	smallint		Minutes of longitude at the grid square.

long_ew	character varying(1)	East or west for longitude at the centre of grid square.
rolgrnd	smallint	Flag for grid square comprising the rolling grounds.
overlandsq	smallint	Flag for square flown when traveling over land.
ema1	smallint	
dlat	numeric(8,5)	Latitude of the grid square in decimal degrees.
dlon	numeric(8,5)	Longitude of the grid square in decimal degrees.
position	geometry	Position of the grid square as gis point type.

```
Indexes:
   "pk_t_grid_code" PRIMARY KEY, btree (code)
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)
```

5.2.9 Table: t_grid_description

```
Comment: Brief descriptions of the half degree squares according to
         coastal and island localities and landmarks.
```

Column	Type	Null?	Description
code	smallint	No	"location" in t_school_sight and "grid" in t_flightpath.
area	character varying(45)		45 character description of half degree square.
area_abbrev	character varying(20)		Abbreviated description of half degree square.

```
Indexes:
   "pk_t_grid_description" PRIMARY KEY, btree (code)
Check constraints:
   "t_grid_description_code_check" CHECK (code >= 0 AND code <= 999)

Foreign-key constraints:
   "fk_t_grid_description" FOREIGN KEY (code)
    REFERENCES aer_sight.t_grid_code(code)
```

5.3 Time tables

5.3.1 Table: t_flt_days

Comment: Various time period labels for days flown (c/f t_time) since the beginning of the database - useful in grouping data.

Column	Type	Null?	Description
flt_grp	smallint	No	Link with five main data tables.
date	date	No	Calendar date.
jul_dy	integer		Julian calendar day (day 2444606 begins at noon on 1/1/1981).
dy_of_yr	smallint	No	Consecutive numbering of days of the year.
dy	smallint		Day of the month.
wk	integer		Week of the year.
yeke	integer		Composite of year and week of the year.
mnth	smallint		Month of the year.
yeth	integer		Composite of year and month of the year.
yr	smallint		Year.

Indexes:
- "pk_t_flt_days" PRIMARY KEY, btree (flt_grp, date)
- "nx_t_flt_days_dy" btree (dy)
- "nx_t_flt_days_dy_of_yr" btree (dy_of_yr)
- "nx_t_flt_days_jul_dy" btree (jul_dy)
- "nx_t_flt_days_mnth" btree (mnth)
- "nx_t_flt_days_wk" btree (wk)
- "nx_t_flt_days_yeke" btree (yeke)
- "nx_t_flt_days_yeth" btree (yeth)
- "nx_t_flt_days_yr" btree (yr)

5.3.2 Table: t_time

Comment: Various time period labels for all days (flown and non-flown) since the beginning of the database to 31/12/95 - useful source for updating t_flt_days.

Column	Type	Null?	Description
date	date	No	Calendar date - link with t_flight_group.
dy_of_yr	smallint	No	Consecutive numbering of days of the year.
jul_dy	integer		Julian calendar day (day 2444606 begins at noon on 1/1/1981).
dy	smallint		Day of the month.
wk	integer		Week of the year.
yeke	integer		Composite of year and week of the year.
mnth	smallint		Month of the year.
yeth	integer		Composite of year and month of the year.
yr	smallint		Year.
fishyr	integer		Fishing year.
fshmnth	smallint		Consecutive numbering (1-12) of months in fishing year (October - September).
qter	smallint		Consecutive numbering (1-4) of annual quarters (Jan-Mar, Apr-Jun, Jul-Sept, Oct-Dec).

Indexes:
- "pk_t_time" PRIMARY KEY, btree (date)
- "nx_t_time_dy" btree (dy)
- "nx_t_time_dy_of_yr" btree (dy_of_yr)
- "nx_t_time_fishyr" btree (fishyr)
- "nx_t_time_jul_dy" btree (jul_dy)
- "nx_t_time_mnth" btree (mnth)
- "nx_t_time_wk" btree (wk)
- "nx_t_time_yeke" btree (yeke)
- "nx_t_time_yeth" btree (yeth)
- "nx_t_time_yr" btree (yr)

5.4 Environmental tables

5.4.1 Table: t_soi

```
Comment: Monthly values of the Southern Oscillation Index (Troup) since
         1/1/67, in units of 0.1 standard deviation.

Column          Type                Null?  Description

yr              smallint              No   Year.

mnth            smallint                   Month - numeric.

soi             numeric(2,1)               Index value.

month1          character varying(3)       Month - alphabetical.


Indexes:
   "nx_t_soi_mnth" btree (mnth)
   "nx_t_soi_month1" btree (month1)
   "nx_t_soi_soi" btree (soi)
   "nx_t_soi_yr" btree (yr)
```

5.4.2 Table: t_astro_gen

Comment: Sun and moon astronomical data.

Column	Type	Null?	Description
jul_dy	integer	No	Julian calendar day (day 2444606 begins at noon on 1/1/1981).
yr	smallint		Year.
mnth	smallint		Month.
dy	smallint		Day of the month.
date	date		Calendar date.
sr0	numeric(5,3)		Sunrise @ the equator; time in decimal hours; value allows for atmospheric refraction.
sr12	numeric(5,3)		Sunrise @ 12 degrees from the equator; time in decimal hours; value allows for atmospheric refraction.
ss0	numeric(5,3)		Sunset @ the equator; time in decimal hours; value allows for atmospheric refraction.
ss12	numeric(5,3)		Sunset @ 12 degrees from the equator; time in decimal hours; value allows for atmospheric refraction.
mr	numeric(5,3)		Moonrise time in decimal hours.
ms	numeric(5,3)		Moonset time in decimal hours.
durn	numeric(5,3)		Duration of the moon above the horizon.
ph	numeric(3,2)		Moon phase.
inten	numeric(5,3)		Intensity of moonlight.
semi	numeric(5,3)		Semi-diameter (radius) of the moon.
decl	numeric(5,2)		Declination of the moon.

Indexes:
"pk_t_astro_gen" PRIMARY KEY, btree (jul_dy)

5.4.3 Table: t_astro_moon

Comment: Reference data on moon cycle since 19/12/79.

Column	Type	Null?	Description
yr	smallint	No	Year.
dy_of_yr	smallint	No	Day of the year.
days_since_epoch	integer	No	Consecutive numbering of days since 19/12/79.
cycle	smallint	No	Consecutive numbering of moon cycles since 19/12/79.
pcent_cycle	numeric(3,2)	No	Percentage of moon cycle elapsed.
quarter	smallint		Moon quarter: 1st, 2nd, 3rd, 4th.

Indexes:
- "pk_t_astro_moon" PRIMARY KEY, btree (days_since_epoch)
- "nx_t_astro_moon_cycle" btree ("cycle")
- "nx_t_astro_moon_dy_of_yr" btree (dy_of_yr)
- "nx_t_astro_moon_pcent_cycle" btree (pcent_cycle)
- "nx_t_astro_moon_quarter" btree (quarter)
- "nx_t_astro_moon_yr" btree (yr)

5.4.4 Table: t_astro_sun

Comment: Reference data on features of the sun cycle.

Column	Type	Null?	Description
mnth	smallint	No	Month.
dy	smallint	No	Day of the month.
up_hr	smallint		Time of sunrise - hour.
up_min	smallint		Time of sunrise - minute.
up_dec	numeric(5,3)		Time of sunrise - decimal hours and minutes.
merid_hr	smallint		Time reaching zenith - hour.
merid_min	smallint		Time reaching zenith - minute.
mer_alt_deg	smallint		Angle to meridian - degrees.
mer_alt_min	smallint		Angle to meridian - minutes.
dn_hr	smallint		Time of sunset - hour.
dn_min	smallint		Time of sunset - minute.
dn_dec	numeric(5,3)		Time of sunset - decimal hours and minutes.
azimuth_deg	smallint		Angle to azimuth.
area	smallint	No	5 degree index areas used by Murray & Burgess 1992 (SBFWS/92/1), and Murray, Taylor, & Vignaux, 1992 (SBFWS/92/2).
dk_hrs	numeric(5,3)		Hours of darkness.

Indexes:
- "pk_t_astro_sun" PRIMARY KEY, btree (area, mnth, dy)
- "nx_t_astro_sun_azimuth_deg" btree (azimuth_deg)
- "nx_t_astro_sun_dk_hrs" btree (dk_hrs)
- "nx_t_astro_sun_dn_dec" btree (dn_dec)
- "nx_t_astro_sun_dn_hr" btree (dn_hr)
- "nx_t_astro_sun_dn_min" btree (dn_min)
- "nx_t_astro_sun_mer_alt_deg" btree (mer_alt_deg)
- "nx_t_astro_sun_mer_alt_min" btree (mer_alt_min)
- "nx_t_astro_sun_merid_hr" btree (merid_hr)
- "nx_t_astro_sun_merid_min" btree (merid_min)
- "nx_t_astro_sun_up_dec" btree (up_dec)
- "nx_t_astro_sun_up_hr" btree (up_hr)
- "nx_t_astro_sun_up_min" btree (up_min)

6 aer_sight business rules

6.1 Introduction to business rules

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

There are three recognised 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 validation.

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.

6.2 Summary of rules

Flight group details (t_flight_group)

flt_grp Flight group number must be an integer greater than zero, and must be unique.

date Must be a valid date and should be greater than 21 Jun 1976.

pilot_code Must be an integer, and be a valid code as listed in the *t_pilot_code* table.

customer_code Must be an integer, and must be a valid code as listed in the *t_customer_code* table.

aircraft_code Must be an integer greater than or equal to zero, and must be a valid code as listed in the *t_aircraft_code* table.

vessel_code Must be an integer greater than or equal to zero and must be a valid code as listed in the *t_vessel_code* table .

Flight details (t_flight)

flt_grp Flight group number must be a valid number as listed in the *t_flight_group* table.

flt_num Must be an integer greater than or equal to zero and should be within the reasonable range of 1 - 25.

Multiple column check on flt_grp and flt_num:
The combination of flt_grp and flt_num must be unique.

flt_indx Must be an integer greater than zero, and be a concatenation of flt_grp and flt_num.

up_field Must be a valid code as listed in *t_airfield_code*.

up_time Must be a valid 24 hour time and fall with in the range of 0 – 2359.

dn_field Must be a valid code as listed in the *t_airfield_code* table.

dn_time Must be a valid 24 hour time and fall with in the range of 0 – 2359.

Multiple column check on up_time and dn_time:
dn_time must be greater than up_time.

hr_min Flight duration in hours and minutes must be a number of 2 decimal places where the decimal can not exceed 0.59.

Multiple column check on up_time and dn_time and hr_min:
The hours and minutes represented by hr_min must be equal to dn_time – up_time.

hr_dec **Multiple column check on up_time and dn_time and hr_dec:**
hr_dec must equal dn_time – up_time in decimal hours.

Sightings details (t_school_sight)

flt_grp	Flight group number must have a value and be a valid number as listed in the *t_flight_group* table.
flt_num	Must be an integer greater than or equal to zero and should be within the reasonable range of 1 - 25.
flt_indx	Must be an integer greater than zero, be a concatenation of flt_grp and flt_num and be a valid flt_indx number in table *t_flight*.
sight_num	Must be an integer greater than or equal to zero and should be within the reasonable range of 1 - 32. **Multiple column check on flt_grp, flt_num, and sight_num:** The combination of flt_grp, flt_num and sight_num must be unique.
sgt_indx	Must be an integer greater than zero. **Multiple column check on flt_grp, flt_indx, and sight_num:** sgt_indx must be a conactenation of flt_grp, flt_indx, and sight_num.
species_code	Must be an integer and be a valid code as listed in *t_species_code*.
num_of_schools	Must be an integer greater than or equal to zero and should be within the reasonable range of 0 - 1000.
ton_min	Must be an integer greater than or equal to zero and should be within the reasonable range of 0 - 1000.
ton_max	Must be an integer greater than or equal to zero and should be within the reasonable range of 1 - 2500. **Multiple column check on ton_min and ton_max:** ton_min must be less than or equal to ton_max.
ton_tot_pil	Must be an integer greater than or equal to zero and should be within the reasonable range of 1 - 15000.
ton_tot_calc	Must be an integer greater than or equal to zero and should be within the reasonable range of 1 - 72000.
sighting_time	Must be a valid 24 hour time in the range 0 – 2359.
time_after_takeoff	Must be a number greater than or equal to zero and should be within the reasonable range of 0.01 - 10.
sea_cond_code	Must be an integer and a valid code in the range 1 – 5 as listed in the *t_sea_cond_code* table.

water_temp	Must be a number, and should be in the range 8 – 30.
location	Must be a valid code as listed in the *t_grid_code* table.
latitude	Must be an integer which represents a valid latitude in the range 30 – 49 degrees. Should be within the range of 33 – 43 degrees.
longitude	Must be an integer which represents a valid longitude in the range 170 – 180 degrees. Should be within the range of 172 to 180.
long_ew	Must be equal to either an 'E' or a 'W'.

Set details (t_set)

flt_grp	Flight group number must have a value and be a valid number as listed in the *t_flight_group* table.
sighting_time	Must be a valid 24 hour time in the range 0 – 2359.
set_time	Must be a valid 24 hour time in the range 0 – 2359.
vessel_code	Must be a valid code in the *t_vessel_code* table.
rst	Must be an integer in the range 0 – 12.
sp_sighting	Must be a valid integer code as listed in the *t_species_code* table.
est_ton_pilot	Must be a number greater than zero and should be within the reasonable range of 1 – 300.
est_sp_pilot	Must be a valid integer code as listed in the *t_species_code* table.
land_ton	Must be a number greater than zero and should be within the reasonable range of 1 – 300.
land_sp	Must be a valid integer code as listed in the *t_species_code* table.

flightpath details (t_flightpath)

flt_grp Flight group number must be a valid number as listed in the *t_flight_group* table.

grid Must have a value and be a valid integer code as listed in the *t_grid_code* table.

ticks Must be an integer or equal to and should be in the reasonable range of 0 - 50.

tic_factor Must be an integer greater than or equal to zero and should be in the reasonable range of 0 - 50.

Pilot codes (t_pilot_code)

code Must have a value, be unique and be an integer greater than or equal to zero.

pilot Pilot name must have a value.

pil_start Must be an integer greater than zero.

Customer codes (t_customer_code)

code Must have a value, be unique and be an integer greater than or equal to zero.

customer Customer name must have a value.

Aircraft codes (t_aircraft_code)

code Must have a value, be unique and be a integer greater than or equal to zero.

aircraft_callsign Must be a multiple of 3 uppercase characters.

Vessel codes (t_vessel_code)

code Must have a value, be unique and be a integer greater than or equal to zero.

vessel_name Must have a value.

num_vess Must be an integer greater than or equal to zero, and should be within the reasonable range of 1 -10.

Airfield codes (t_airfield_code)

code Must have a value, be unique and be a integer greater than or equal to zero.

Species codes (t_species_code)

code Must have a value, be unique and be an integer greater than or equal to zero.

kah_mix Must be an integer and should have a value of 0 or 1.

jma_mix Must be an integer and should have a value of 0 or 1.

ema_mix Must be an integer and should have a value of 0 or 1.

skj_mix Must be an integer and should have a value of 0 or 1.

tre_mix Must be an integer and should have a value of 0 or 1.

bma_mix Must be an integer and should have a value of 0 or 1.

jma_grp Must be an integer and should have a value of 0 or 1.

Sea conditions codes (t_sea_cond_code)

code Must have a value, be unique and be an integer in the range 1 – 4.

old_sea_cond Must have a value (of type character).

old_to_new Must be an integer and should be in the range 1 – 3.

Grid codes (t_grid_code)

code	Must be an integer, have a value, be unique and should be in the range 0 – 445.
qmaarea	Must be an integer greater than zero and should be in the range 1 - 10.
local_area	Must be an integer and should be in the range 1 – 6.
tod1	Must be an integer and should be in the range 1 – 4.
kah_ma	Must be an integer and should be in the range 1 - 9.
jma_ma	Must be an integer and should be in the range 1 - 9.
tre_ma	Must be an integer and should be in the range 1 - 9.
fishery	Must be an integer and should be in the range 1 – 2.
lat_deg	Must be an integer and should be in the range 33 – 42.
lat_min	Must be an integer and be in the range 0 – 59.
long_deg	Must be an integer and should be in the range 172 – 180.
long_min	Must be an integer and be in the range 0 – 59.
long_EW	Must be equal to either an ‘E’ or a ‘W’.
rolgrnd	Must be an integer and should be either null or equal to 1.
overlandsq	Must be an integer and should be either null or equal to 1.

Grid descriptions (t_grid_description)

code	Must have a value, be unique and be a integer in the range 0 – 999.

Flight days time details (t_flt_days)

flt_grp	Flight group number must be a valid number as listed in the *t_flight_group* table.
date	Must have a value and be a valid date.
jul_day	Must be a valid Julian date.
dy_of_yr	Must have a value and be an integer in the range 1 – 366.
dy	Must be an integer and be in the range 1 – 31.
wk	Must be an integer and be in the range 1 – 53.
yeke	Must be an integer greater than or equal to 197625 and be a combination of a valid 4 digit year and a valid week of year.
mnth	Must be an integer in the range 1 – 12.
yeth	Must be an integer greater than or equal to 197606 and must be a combination of a valid 4 digit year and a valid month of year.
yr	Must be an integer greater than zero and should be in the range 1976 to the current year.

Date and time data (t_time)

date	Must have a value and be a valid date.
dy_of_yr	Must have a value and be an integer in the range 1 – 366.
jul_day	Must be a valid Julian date.
dy	Must be an integer and be in the range 1 – 31.
wk	Must be an integer and be in the range 1 – 53.
yeke	Must be an integer greater than or equal to 197625 and be a combination of a valid 4 digit year and a valid week of year.
mnth	Must be an integer in the range 1 – 12.
yeth	Must be an integer greater than or equal to 197606 and must be a combination of a valid 4 digit year and a valid month of year.
yr	Must be an integer greater than zero and should be in the range 1976 to the current year.
fishyr	Must be an integer greater than zero where the first 4 digits represent the start calendar year of the fishing year followed by the last 2 digits of the next calendar year.
fishmnth	Must be an integer in the range 1 – 12.
qter	Must be an integer in the range 1 – 4.

Southern Oscillation Index data (t_soi)

yr Must be an integer, have a value, and be in the range 1967 to the current year.

mnth Must be an integer in the range 1 – 12.

month1 Must be a 3 character lowercase code for a month as listed below: 'jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec'

Astronomical data (t_astro_gen)

jul_day Must be a valid Julian date.

yr Must be an integer greater than zero and should be in the range 1976 to the current year.

mnth Must be an integer in the range 1 – 12.

dy Must be an integer and be in the range 1 – 31.

date Must be a valid date.

sr0, sr12, ss0, ss12, mr, ms, durn Must be a number in the range 0 – 23.999.

durn Must be a number and should be in the range 0 – 13.9.

ph Moon phase must be a number in the range 0 – 1.0.

inten Must be a number and should be in the range 0 – 1.0.

semi Must be a number in the range 0.490 – 0.559.

decl Must be a number in the range –28.68 – 28.68.

Moon cycle data (t_astro_moon)

yr Must have a value, be an integer greater than zero and should be in the range 1979 to the current year.

dy_of_yr Must have a value and be an integer in the range 1 – 366.

days_since_epoch Must have a value and be an integer greater than or equal to zero.

cycle Must have a value and be an integer greater than or equal to zero.

pcent_cycle Must have a value and be in the range 0 – 0.99.

quarter Must be an integer in the range 1 – 4.

Sun cycle data (t_astro_sun)

mnth	Must have a value and be an integer in the range 1 – 12.
dy	Must have a value and be an integer in the range 1 – 31.
up_hr	Must be an integer greater than zero and should be in the range 3 – 9.
up_min	Must be an integer and be in the range 0 – 59.
up_dec	Must be a positive number and should be in the range 4.0 – 8.9.
merid_hr	Must be an integer greater than zero and should be in the range 10 – 14.
merid_min	Must be an integer and be in the range 0 – 59.
mer_alt_deg	Must be an integer and should be in the range 18 – 76.
mer_alt_min	Must be an integer and be in the range 0 – 59.
dn_hr	Must be and integer greater than zero and should be in the range 15 – 21.
dn_min	Must be an integer and be in the range 0 – 59.
dn_dec	Must be a positive number and should be in the range 16.0 – 20.9.
azimuth_deg	Must be an integer greater than zero and should be in the range 54 – 128.
area	Must have a value, be an integer and should be in the range of 1- 4.
dk_hrs	Must be a positive number and should be in the range 8.0 – 15.9.

7 Acknowledgements

I would like to thank Kevin Mackay and Brian Sanders and for their technical review and editorial comment on this document, and Peter Shearer for producing the table listings.

8 References

Bradford, E. and Taylor, P.R. 1995. Trends in pelagic fish abundance from aerial sightings data. *New Zealand Fisheries Assessment Research Document 9518.*

Ng, S. 1992. Standards for setting databases and their applications. *MAF Fisheries Greta Point Internal Report No. 180.* 31 p. (Report held at MAF Fisheries Greta Point Library, Wellington).

Taylor, P.R. in press. Exploratory analysis of aerial sightings of pelagic schooling species, and a method for estimating relative abundance indices. *Draft NIWA Technical Report.*

Appendix 1 - Data Integrity

Data collection and data processing.

Data are collected on the standardised forms, which are shown in Appendix 3.
Books of forms are provided by the Pelagic Research Group of NIWA (i.e., the aerial sightings database manager), are filled out by the pilots and returned to Greta Point. For a full discussion of this topic see Taylor *(in preparation).*

Data entry, process, and definitions.

This section outlines the flow of paper recorded data, for aerial sightings data from field collection through to its availability to researchers for stock assessment analyses, and defines the separate tasks that are required to do this.

In this example pilots flying light aircraft collect hand written data. These data are recorded on paper forms.

At the completion of each flight the recorder ensures that all pages are in order and that all required data fields have been correctly filled. The data are then forwarded to a project team member (the client e.g., NIWA) responsible for checking the data prior to keypunching.

There are 5 clear steps in the data flow following its collection. These are listed and then discussed individually in detail.
1. Pre-key punching, checking and batching.
2. Key punching data entry.
3. Electronic transfer of raw data flat files in disk and paper copy to client (i.e., project team).
4. Data error checking (manual and computer), validation, and grooming.
5. "Groomed", validated data loaded to database. Now available for analysis.

1. **Pre-key punching, visual checking and batching:**
The paper forms from each flight are checked for obvious gross errors or omissions and corrected if necessary. Forms are placed into batches and allocated a unique file name. The batches of raw data are sent for keypunching.

2. **Key punching data entry:**
At keypunching, the batches of raw data are digitised and verified by trained data entry operators. Verification simply means that the data are digitised twice and the two resulting files are crosschecked for mismatches. Operator errors are corrected at this point, and the completed digitised file is ready for transfer for the error checking process. At no point in this process are changes or interpretations made to the raw data. NIWA uses the KEYS Data Emulator for data entry.

3. **Electronic transfer of raw data flat files in disk and paper copy to client:**
The digitised data file is transferred for error checking, along with the original raw data file. At this point the data are now in a format that is compatible with the data processing routines.

4. **Data error checking, validation, and grooming:**
Data files are put through a number of computer error checking (validation) routines that look for inaccuracies and inconsistencies in the data. Any errors detected are corrected. Data are then passed through these error-checking routines until the data reach a satisfactory standard that will allow them to be inserted in the appropriate database.

In some instances, data may be inserted into "working tables" in a database. This is often done to check the integrity of the data by taking advantage of relational databases ability to manipulate, match, and compare related sets of data. Details for this aerial sightings data are given below.

a. Reformatting raw data files (file_1) using the programme "modify-raw" in the directory "/neptune/grp1/sightings/aer_sigh_wd/dat_load_wd/chek_progs" - this programme calculates some estimates of search effort and sightings tonnages, adds some extra attributes which increase the efficiency of links between tables and writes to an output file (file_2) in the appropriate format for running through the error checking programmes;

b. Error checking using the checkq programme "as_check" in the same directory as above - this programme checks file_2 for values out of range, for null values etc. and does some minor formatting including insertion of field delimiters before producing five output files, one for each of the main database tables (flight_group, flight, school_sight, set, flightpath), and one (as_errors) which contains error messages for the input file;

c. Using as-errors, file_1 is then edited and steps 1 and 2 repeated - until satisfied that the errors have been corrected.

5. **"Groomed", validated data loaded to database. Available for analysis:**
The files of clean, groomed, and validated data are inserted into the appropriate database and now become available for analysis.

The clean digitised data files and raw paper data are then archived for safekeeping.