DIASPARA WP3.4 FISH STOCK DATABASE : METRIC DB final report

version = pre-release, for review. Still have to integrate salmon from WP2

Briand Cédric, Oliviéro Jules, Helminen Jani

January 16, 2026

Co-funded by the European Union
The EU is not responsible for the content of the project

Contents

Note : If you are reading this file as a pdf you should try to open it on the diaspara website, as some code does not render well as a pdf : stock database

List of Tables

Despite legal commitments for their conservation, the Atlantic salmon and the European eel are currently endangered. This is partly due to their ecological characteristics. First, the two species share their life cycle between marine and continental ecosystems, in and outside Europe. Despite behaving like independent units during their continental phase, they are biologically mixed during their marine phases, requiring to orchestrate regional and international management and assessment process (data collection and availability, use of appropriate assessment methods). Moreover, the species are submitted to many human impacts (e.g. fisheries, habitat degradation and fragmentation). In this context, building on pre-existing road maps, DIASPARA aims to provide tools to enhance the coherence of the scientific assessment process from data collection to assessment, with the final objective of supporting more holistic advice and to better inform a regional management. In the second work package, DIASPARA has done an inventory of available data and made recommendations for potential improvement in the collection, based on a spatiotemporal analysis of key biological parameters. In the database work package (WP3 - this current WP), DIASPARA aimed to develop database structures in order to store data required for the assessment. This was created to include biological data and fisheries data, but also data to monitor the impact of dams and hydropower plants.

Currently, both ICES WGEEL (Working group on eel) and WGNAS (Working group on North Atlantic Salmon) have developed “home-made” databases stored in local institutions, alongside interactive applications to explore and integrate the data. WGBAST (Working group on the Baltic Atlantic Salmon and Sea Trout) relies on a very extensive set of tables collated in excel to run various models. These approaches are far from optimal in terms of operability, data availability, data security and transparency. Moving towards a transparent assessment framework (TAF) procedure requires simpler and more transparent ways of querying several central databases to get the raw data and do the assessment. The objective of this WP is to create database structures to store data to feed models that are currently in use, as well as data that will be useful in the future to support a holistic and transparent assessment.

The first part of the work has been to exchange with the different working groups

and analyze the content of their databases as well as their working processes.

The reports are available for WGNAS, WGBAST, WGTRUTTA. WGEEL database was developed largely by the leaders of this work package, and was not described.

The second part consisted of finding a database structure suiting the needs of all expert groups. The main structure of the database has been proposed during the online DIASPARA meeting : database structure.

This structure allows to store the data necessary to run the international stock models for the different working groups giving advice or scientific reports on diadromous fishes. The structure is similar to both the WGEEL and the WGNAS databases.

The third part is creating referential table for the different types proposed, and then populates the database with the contents of the WGEEL and WGNAS database. All chunks have been run sequentially to create the database, but some are later marked with eval = FALSE to speed up the process of running the quarto document.

Finally the fourth part is to import data in the stock database and to create a structure working for all the working groups.

The diadromous fish database is in fact made of two main structures. The first is called ***Stock Database*** and holds values aggregated at the scale of the assessment unit, or stock unit. The second is the ***Metric Database*** and allows to have time series of data, group and individual metrics to store life history traits data. The report for the metric can found here.

This habitat used to support both structures has been created using a hierarchical structure of the habitat of the migratory fishes (see habitat report). The script to create this report can be found in diaspara github code of the report, the SQL codes are found in the following link SQL code.

Chapter 1

Hierarchical structure of the database

The database starts with the simplest structure, with a basic table corresponding to all data. Then three tables are created, one per Working group Since SQL server does not handle inheritance, once the table built, some of those will have to be replaced with views or partitioned views.

> **i** Code for SQL Server
> SQL server used by ICES does not use inheritance, the same structure (schema) can be followed and the mother table will correspond to a view either as a UNION or JOIN of the tables

1.1 Referential tables

Similarly, referential tables are created with a mother table from which specific (or WG-specific) tables inherit. All mother tables will be held in a schema called `ref`. Having working-group-specific tables makes setting consistent foreign key easier. For instance, WGEEL references different stages than WGNAS. Most of these referential tables are common between species, and whenever possible, they are sourced from ICES vocab.

1.2 Unicity constraints

Another important point to add is unique constraint. As some values would be null, creating unique constraints with indexes would be necessary. These allow to have different levels of constraints for instance the unique constraint would be defined for : (`year, age, area, parameter`) (`year, age, parameter`) (`year,`

`area, parameter`) (`year, parameter`). Age is used by WGNAS and WGBAST but not WGEEL, and WGBAST has two additional fields not found in the other databases: period (month, half of year, …), with the value and the estimation method. Using different structures in the inherited table is a way to deal with the differences between working groups while keeping most of the structure common.

In WGNAS because of the presence of matrix (area x area) the area might appear twice, this will be dealt with the use of an additional column (both for WGBAST and WGNAS). There might be a need for a trigger to check the unicity (`year, area, area, parameter`)

1.3 Creating the diaspara database

Along this document, the database will be named `DiadromousDB`. The database is created with postgresSQL.

Some rules

- By default, values are be capitalised for the first letter e.g. `Public` for the code in dataaccess.
- Codes are capitalized, e.g working group names WGEEL, WGNAS, or country codes.
- Units are lowercase e.g. g, mm …
- Column naming: all integer used for primary keys are called `xxx_id` all columns containing text used for primary keys are called `xxx_code`, the column with definition, or description is always called `xxx_description` where xxx is the three letter code for the table.
- All tables end with a 3 letter summary which allows to identify the source of one column so the table dataaccess will be called `tr_dataaccess_dta`. And the column for code will be named.
- Referential tables or dictionaries are called `tr_`(sometable), tables build in conjuction from other tables and not in the dictionaries are called `t_`(sometable).
- Foreign keys are used instead of check constraints as check constraint might not ensure the integrity of data after their integration (only when new rows are created or modified).
- Foreign keys are name `fk_columnname` where the column name is the name in the table
- Primary keys are names `tablename_pkey` (with the constraint possibly referering to more than one column).
- Other constraints are check constraints `ck_columnname` and unique contraints `uk_columnname`

- All tables and column have a definition, we will ask the working groups to check those.
- Use of "snake_case" : Column and table name are ALWAYS LOWERCASE, the underscore is only used to separate type of table and table shortcode t_table_abc. In column is separates table code abc_def_code (table abc will reference the column def_code in table def). Some exceptions to this rule are made when the table was imported straight from ICES

> Another code in ICES
> ICES uses CamelCase and not snake_case, but using upper case in postgres is difficult, and requires the use of double quotes, when using SQL strings in R it then becomes very difficult to write SQL. So we followed a more postgres compatible case here. This will change once the format will be created in ICES.

The database can be created an run in localhost, check the wp3_habitat repository for code to set up access to the database. Two roles are created, `diaspara_admin` and `diaspara_read` and users are given specific rights.

> DIASPARA technical note
> When installing diaspara on a server with external connection, there is a need to edit the pb_hba.conf on the server if not in localhost to allow access to diaspara.

```
# dbExecute(con_diaspara_admin, "DROP schema if exists ref CASCADE;");
# dbExecute(con_diaspara_admin, "CREATE schema ref;")
# dbExecute(con_diaspara_admin, "GRANT ALL PRIVILEGES ON SCHEMA ref TO diaspara_admin ;")
# dbExecute(con_diaspara_admin, "GRANT ALL PRIVILEGES ON SCHEMA public TO diaspara_read ;")
# dbExecute(con_diaspara_admin, paste0("GRANT CONNECT ON DATABASE ",cred$dbnamediaspara," TO
# dbExecute(con_diaspara_admin, paste0("ALTER DATABASE ",cred$dbnamediaspara," OWNER TO dias
# dbExecute(con_diaspara_admin, "DROP schema if exists refeel CASCADE;");
# dbExecute(con_diaspara_admin, "CREATE SCHEMA refeel;")
# dbExecute(con_diaspara_admin, "ALTER SCHEMA refeel OWNER TO diaspara_admin;")
# dbExecute(con_diaspara_admin, "DROP schema if exists refnas CASCADE;");
# dbExecute(con_diaspara_admin, "CREATE SCHEMA refnas;")
# dbExecute(con_diaspara_admin, "ALTER SCHEMA refnas OWNER TO diaspara_admin;")
# dbExecute(con_diaspara_admin, "DROP schema if exists refbast CASCADE;");
# dbExecute(con_diaspara_admin, "CREATE SCHEMA refbast;")
# dbExecute(con_diaspara_admin, "ALTER SCHEMA refbast OWNER TO diaspara_admin;")
# dbExecute(con_diaspara_admin, "DROP schema if exists reftrutta CASCADE;");
# dbExecute(con_diaspara_admin, "CREATE SCHEMA reftrutta;")
# dbExecute(con_diaspara_admin, "ALTER SCHEMA reftrutta OWNER TO diaspara_admin;")
#
```

```
# # Create foreign data wrapper to wgeel database
#
# dbExecute(con_diaspara_admin, "CREATE EXTENSION IF NOT EXISTS postgres_fdw;")
#
# dbExecute(con_diaspara_admin,"
# CREATE SERVER wgeel_data_wrapper
#   FOREIGN DATA WRAPPER postgres_fdw
#   OPTIONS (host 'localhost', port '5432', dbname 'wgeel');")
# dbExecute(con_diaspara_admin,"
# CREATE SERVER wgnas_data_wrapper
#   FOREIGN DATA WRAPPER postgres_fdw
#   OPTIONS (host 'localhost', port '5432', dbname 'salmoglob');")
# dbExecute(con_diaspara_admin,"
# CREATE USER MAPPING FOR USER
#   SERVER wgeel_data_wrapper
#   OPTIONS (user 'postgres', password 'postgres');")
# dbExecute(con_diaspara_admin,"
# CREATE SCHEMA refwgeel;")
# dbExecute(con_diaspara_admin,"IMPORT FOREIGN SCHEMA ref
#     FROM SERVER wgeel_data_wrapper
#     INTO refwgeel;")
#
# dbExecute(con_diaspara_admin, paste0("COMMENT ON DATABASE ",cred$dbnamediaspara," IS 'This
# dbExecute(con_diaspara_admin,
#           "GRANT ALL PRIVILEGES ON SCHEMA refwgeel TO diaspara_admin;")
```

Now the database has been created with different schemas (Figure 1.1). The main schema for dictionaries is ref, and a schema is created per working group for specific referential tables. The Schema refwgeel has been filled in with a foreign data wrapper to get the data from wgeel, the same schema exists for wgnas. We'll see later for wgbast and wgtrutta. The schema `dat` is the common schema for all data. For each working group, schema `datbast`, `dateel`, `datnas` are created. The tables are be created in dat and later similar or a bit more complex tables (with some more columns) will be created using the `INHERIT FROM` syntax, which will allow to have a hierarchical structure in the db, and maintain the structure in a table common to all fishes. Note that `dat` should not containt any data, but will hold all the views and inherited tables coming from the different schema.

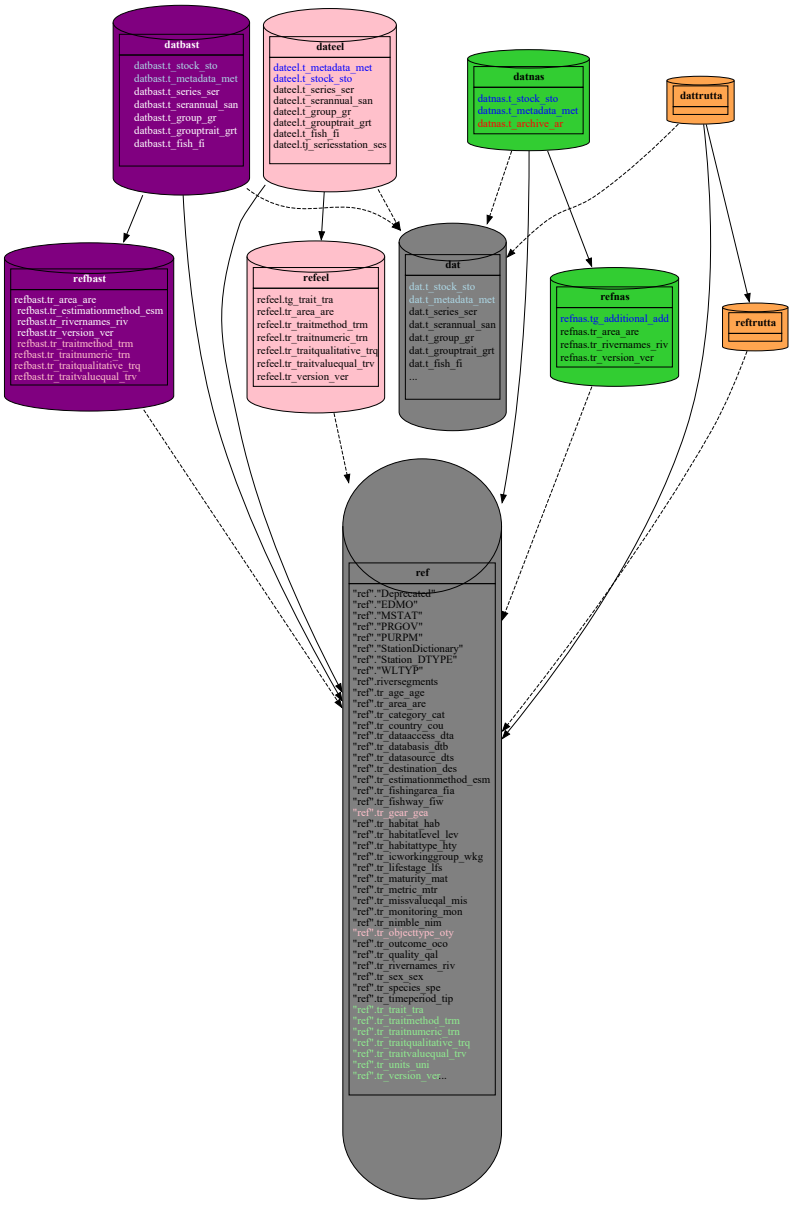


Figure 1.1: Conceptual schema of the diadromous database

Chapter 2

Creating referentials

This script holds all referentials necessary for both the metricDB and the stockDB.

2.1 species (tr_species_spe)

The first thing is to create a referential table for species. At the moment the structure doesn't integrate schema for Alosa and Lamprey, but the species have been created. There are no codes in ICES vocab for **Alosa alosa**, **Alosa fallax**, **Petromyzon marinus**, **Lampetra fluviatilis**.

QUESTION to ICES: species codes

ANG, ALA, ALF, PET, LAM are these internal codes OK? Should we use SpecWoRMS or is Aphia OK?

ANSWER ICES: Maria

For species, we would recommend that you use AphiaIDs (a copy of which is SpecWoRMs). You can also use the FAO ASFIS list, or both, but we would recommend having the AphiaIDs for sure. => DONE

ANSWER WGTRUTTA : Iain

I would suggest the common name for Salmo Trutta should just be trout (as you normally can't differentiate migratory and resident forms of the juveniles). In our database we also have a field "salmonid" for circumstances where people electrofish very early in the year and it isn't readily possible to separate trout and salmon fry.

```
sp <- getCodeList("IC_species")

#The following lines show that there is no code in IC_species
#grep("Lampetra", sp$description) # nothing
```

```
#grep("Petromyzon", sp$description) # nothing
#grep("Alosa",  sp$description) # nothing

bind_rows(
  ele <- getCodeDetail("IC_species","ELE")$detail,
  sal <- getCodeDetail("IC_species","SAL")$detail,
  trs <- getCodeDetail("IC_species","TRS")$detail) |>
  knitr::kable(caption = "Codes for migratory species in ICES, no code found for other speci
```

Table 2.1: Codes for migratory species in ICES, no code found for other species (Lamprey, Alosa …)

Id	Guid	Key	Description	LongDescription	Modified
77905	fea31ebb-fc69-4562-af1b-3fec866e7e58	ELE	Anguilla anguilla		2019-04-15T
77934	810c0c92-d333-4b16-ab8c-dfe63a7c1a20	SAL	Salmo salar		2019-04-15T
77943	b0de7924-ee6c-483e-a2e3-91c80ca033c2	TRS	Salmo trutta		2019-04-15T

```
if (file.exists("data/tr_species_spe_temp.Rdata")) {
  load("data/tr_species_spe_temp.Rdata") } else {
    species_list <- tibble(
      spe_code = c("127186", "126281", "127187", "126413", "126415", "101174", "101172"),
      spe_icspecieskey = c("SAL", "ELE", "TRS", NA,NA,NA,NA),
      spe_commonname = c("Atlantic salmon", "European eel", "Sea trout", "Twait shad", "Alli
      spe_scientificname = c("Salmo salar", "Anguilla anguilla", "Salmo trutta", "Alosa alos
    )
    tr_species_spe_temp <- species_list |>
      rowwise() |>
      mutate(
        spe_codeaphia = findAphia(spe_scientificname, latin = TRUE)
      ) |>
      ungroup()
    save(tr_species_spe_temp, file = "data/tr_species_spe_temp.Rdata")
  }
knitr::kable(tr_species_spe_temp) |> kable_styling(bootstrap_options = c("striped", "hover",
```

spe_code	spe_commonname	spe_scientificname	spe_codeaphia
SAL	Atlantic salmon	Salmo salar	127186
ELE	European eel	Anguilla anguilla	126281
TRT	Sea trout	Salmo trutta	127187
ALA	Twait shad	Alosa alosa	126413
ALF	Allis shad	Alosa fallax	126415

SLP	Sea lamprey	Petromyzon marinus	101174
RLP	European river lamprey	Lampetra fluviatilis	101172

SQL code to create table `tr_species_spe`

```
--DROP TABLE IF EXISTS ref.tr_species_spe;
CREATE TABLE ref.tr_species_spe (
     spe_code CHARACTER VARYING(3) PRIMARY KEY,
     spe_commonnname TEXT,
     spe_scientificname TEXT,
     spe_codeaphia numeric NOT NULL,
     spe_description TEXT);
COMMENT ON TABLE ref.tr_species_spe IS 'Table of species code';
GRANT ALL ON TABLE ref.tr_species_spe to diaspara_admin;
GRANT SELECT ON TABLE ref.tr_species_spe to diaspara_read;

-- 19/12/2025 finally we need TO use AFIAID
-- we also remove TRT as a species

DELETE FROM "ref".tr_species_spe  WHERE spe_code='TRT';


ALTER TABLE ref.tr_species_spe ALTER column spe_code type TEXT;


ALTER TABLE "ref"."tr_lifestage_lfs" ALTER COLUMN lfs_spe_code TYPE TEXT;

-- 19/12/2025 finally we need TO use AFIAID
-- we also remove TRT as a species

ALTER TABLE dat.t_metadata_met ALTER COLUMN met_spe_code TYPE TEXT;

ALTER TABLE dat.t_stock_sto ALTER COLUMN sto_spe_code TYPE TEXT;

 ALTER TABLE datnas.t_metadata_met DROP
    CONSTRAINT ck_met_spe_code;

 ALTER TABLE datnas.t_stock_sto DROP CONSTRAINT
ck_spe_code;

ALTER TABLE datbast.t_metadata_met DROP CONSTRAINT ck_met_spe_code;

ALTER TABLE dateel.t_metadata_met DROP CONSTRAINT ck_met_spe_code;

ALTER TABLE datnas.t_stock_sto DROP CONSTRAINT ck_spe_code;
```

```
ALTER TABLE dateel.t_stock_sto DROP CONSTRAINT ck_spe_code;

ALTER TABLE datbast.t_stock_sto DROP CONSTRAINT ck_spe_code;

UPDATE ref.tr_species_spe SET spe_code = spe_codeaphia;


ALTER TABLE datnas.t_metadata_met ADD
    CONSTRAINT ck_met_spe_code
    CHECK (met_spe_code='127186');

ALTER TABLE datnas.t_stock_sto ADD CONSTRAINT
ck_sto_spe_code CHECK (sto_spe_code='127186');

ALTER TABLE datbast.t_metadata_met
ADD CONSTRAINT ck_met_spe_code CHECK
 (met_spe_code='127186' OR met_spe_code='127187');

 ALTER TABLE datbast.t_stock_sto ADD CONSTRAINT
ck_sto_spe_code CHECK  (sto_spe_code='127186' OR sto_spe_code='127187');

ALTER TABLE dateel.t_metadata_met
ADD CONSTRAINT ck_met_spe_code CHECK (met_spe_code='126281');

ALTER TABLE datbast.t_stock_sto
ADD CONSTRAINT ck_spe_code
CHECK (sto_spe_code='127186' OR sto_spe_code='127187');

COMMENT ON COLUMN datbast.t_metadata_met.met_spe_code
IS 'Species aphiaID, text ''127186'' salmo salar OR ''127187'' for Salmo trutta primary key

COMMENT ON COLUMN dateel.t_metadata_met.met_spe_code
IS 'Species, ''126281'' primary key on both met_spe_code and met_var.';

ALTER TABLE dateel.t_stock_sto ALTER COLUMN sto_spe_code SET DEFAULT '126281';
ALTER TABLE datnas.t_stock_sto ALTER COLUMN sto_spe_code SET DEFAULT '127186';
-- no default for datbast

COMMENT ON TABLE  ref.tr_species_spe IS
'Table of fish species, spe_code using AphiaID as the reference with
reference to ICES vocabularies.'
```

```
dbWriteTable(conn=con_diaspara, name = "tr_species_spe_temp", value = tr_species_spe_temp, o
dbExecute(con_diaspara,"INSERT INTO ref.tr_species_spe SELECT * FROM tr_species_spe_temp")#7
dbExecute(con_diaspara,"DROP TABLE tr_species_spe_temp")
```

```
dbExecute(con_diaspara_admin, "COMMENT ON TABLE  ref.tr_species_spe IS
'Table of fish species, spe_code using AphiaID as the reference with
reference to ICES vocabularies.'")
```

Table 2.3: Working groups table in the diaspara DB

spe_code	spe_commonnname	spe_scientificname	spe_codeaphia	spe_description
127186	Atlantic salmon	Salmo salar	127186	NA
126413	Twait shad	Alosa alosa	126413	NA
126415	Allis shad	Alosa fallax	126415	NA
101174	Sea lamprey	Petromyzon marinus	101174	NA
101172	European river lamprey	Lampetra fluviatilis	101172	NA
126281	European eel	Anguilla anguilla	126281	NA
127187	Sea trout	Salmo trutta	127187	NA

2.2 Working group (tr_icworkinggroup_wkg)

Species is necessary to separate data within the same working group (WGBAST works on both Trutta and Salmon). Furthermore, two working groups might be working on the same species. For this reason we need to have a "working group" entry in most of the tables. There is already a table for working group. Proposed table is Table 2.4 but need confirmation for WKTRUTTA2.

> **Note**
>
> WGEEL name needs to be updated to JOINT EIFAAC/ICES/GFCM WORKING GROUP ON EEL and WGTRUTTA is an expert group, and, therefore, needs to be added to the lit separately.
> The working groups might change over time, referencing a working group there is probably not the best. > Added stockkeylabel, this table is necessary to aggregate data, as species is not enough.

SQL code to create table `tr_icworkinggroup_wkg`

```
--DROP TABLE IF EXISTS ref.tr_icworkinggroup_wkg CASCADE;

CREATE TABLE ref.tr_icworkinggroup_wkg (
wkg_code TEXT PRIMARY KEY,
wkg_description TEXT,
wkg_icesguid uuid,
wkg_stockkeylabel TEXT
);
```

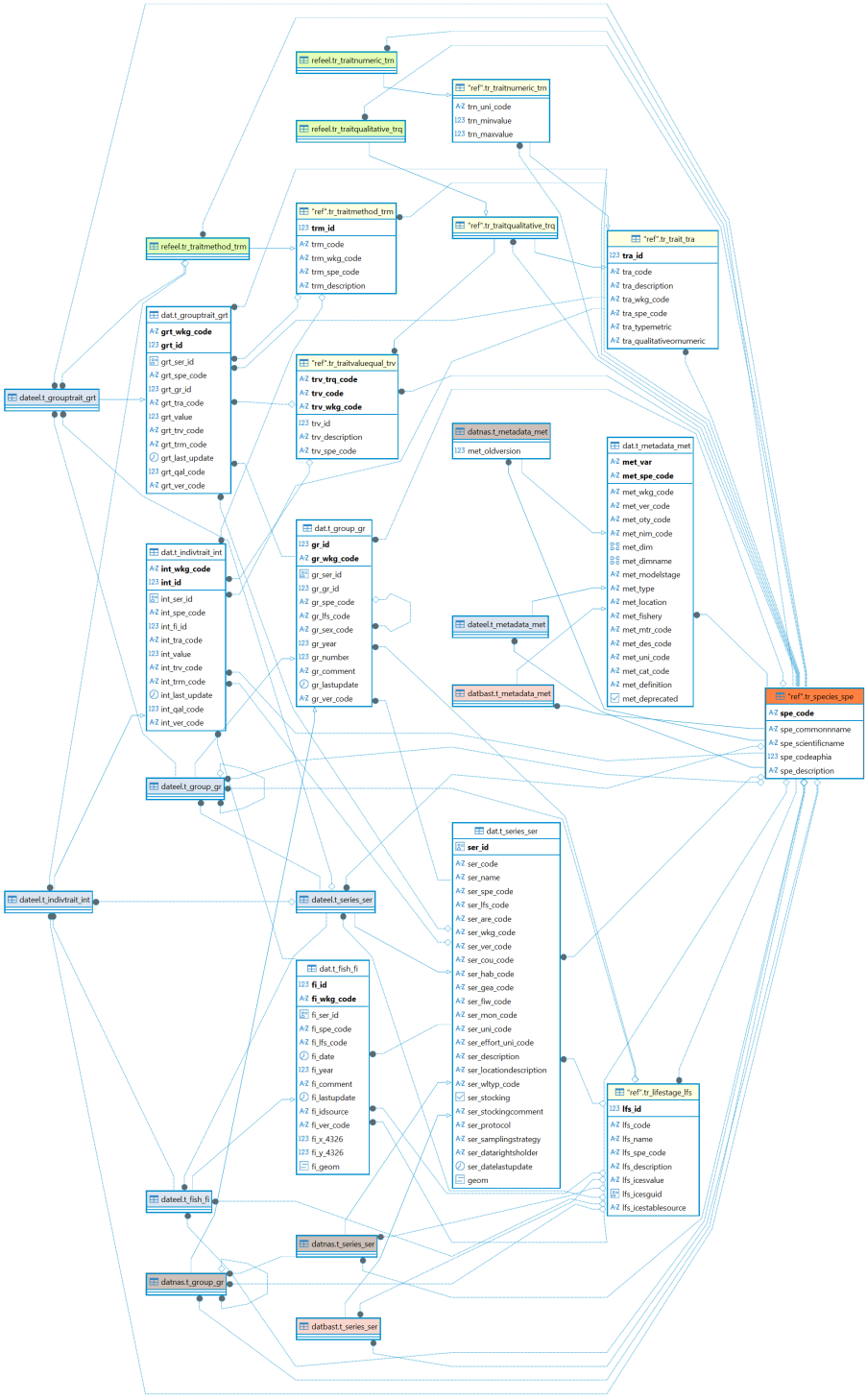


Figure 2.1: diagram of tr_species_spe

```
COMMENT ON TABLE ref.tr_icworkinggroup_wkg
IS 'Table corresponding to the IC_WorkingGroup referential;';
COMMENT ON COLUMN ref.tr_icworkinggroup_wkg.wkg_code IS
'Working group code uppercase, WGEEL, WGNAS, WGBAST, WGTRUTTA';


GRANT ALL ON ref.tr_icworkinggroup_wkg TO diaspara_admin;
GRANT SELECT ON ref.tr_icworkinggroup_wkg TO diaspara_read;
```

```
# Using the jsonlite to download the guid also
tbl <- jsonlite::fromJSON(
  "https://vocab.ices.dk/services/api/Code/3f6fb38a-a3c5-4f5c-bf31-2045e12536ee")



temp_tr_icworkinggroup_wkg <- tbl |>
  select(key,description,guid) |>
  rename(wkg_code = key,
         wkg_description = description,
         wkg_icesguid = guid)|>
  filter(wkg_code %in% c("WGEEL", "WGNAS", "WGBAST"))
temp_tr_icworkinggroup_wkg <- bind_rows(temp_tr_icworkinggroup_wkg,
                                        data.frame(wkg_code="WKTRUTTA"))
temp_tr_icworkinggroup_wkg$wkg_stockkeylabel <-
  c("sal.27.22-31","ele.2737.nea","sal.neac.all",NA)
dbWriteTable(con_diaspara_admin,
             "temp_tr_icworkinggroup_wkg",
             temp_tr_icworkinggroup_wkg,
             overwrite = TRUE)

dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_icworkinggroup_wkg
          SELECT
          wkg_code,
          wkg_description,
          wkg_icesguid::uuid,
          WKG_stockkeylabel
         FROM temp_tr_icworkinggroup_wkg;") #4
dbExecute(con_diaspara_admin, "DROP TABLE temp_tr_icworkinggroup_wkg;")
```

Table 2.4: Working groups table in the diaspara DB

wkg_code	wkg_description	wkg_icesguid
WGBAST	Baltic Salmon and Trout Assessment Working Group	2cac261d-c837-459a-961b-e63e36cc1
WGEEL	Joint EIFAAC/ICES Working Group on Eels	7c13a79e-7855-4d0b-b567-21300bcaa
WGNAS	Working Group in North Atlantic Salmon	b5fd158e-b153-4e2e-a6da-c4b0536d6

WKTRUTTA	NA	NA

2.3 Country (tr_country_cou)

Countries are taken from the wgeel database where everything is almost OK and streamlined with ICES. But we need to add North American countries. The shapefiles have been downloaded from https://gisco-services.ec.europa.eu/distribution/v2/countries/download/#countries source EuroGeographics and UN-FAO. Countries (Table 2.5) are ordered from North to South starting from the Baltic and ending in the Mediterranean, with American number being the highest in order.

SQL code to create table `tr_country_cou`

```
DROP TABLE IF EXISTS ref.tr_country_cou;
CREATE TABLE ref.tr_country_cou (
    cou_code character varying(2) NOT NULL,
    cou_country text NOT NULL,
    cou_order integer NOT NULL,
    geom public.geometry,
    cou_iso3code character varying(3)
);
COMMENT ON TABLE ref.tr_country_cou IS
          'Table of country codes source EuroGeographics and UN-FAO.';
ALTER TABLE ref.tr_country_cou
          OWNER TO diaspara_admin;
GRANT SELECT ON TABLE ref.tr_country_cou TO diaspara_read;
```

```
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_country_cou
          SELECT * FROM refwgeel.tr_country_cou;") #40
# Add some constraints
dbExecute(con_diaspara_admin, "ALTER TABLE ref.tr_country_cou
          ADD CONSTRAINT t_country_cou_pkey PRIMARY KEY (cou_code);")
dbExecute(con_diaspara_admin, "ALTER TABLE ref.tr_country_cou
          ADD CONSTRAINT uk_cou_iso3code UNIQUE (cou_iso3code);")

# missing values from America downloaded from https://gisco-services.ec.europa.eu/distributi
# uploaded to postgres

# the tables

# ref-countries-2024-01m - CNTR_RG_01M_2024_4326
# have been copied to folder area ref-countries was renamed
# ALTER TABLE area."ref-countries-2024-01m - CNTR_RG_01M_2024_4326"
```

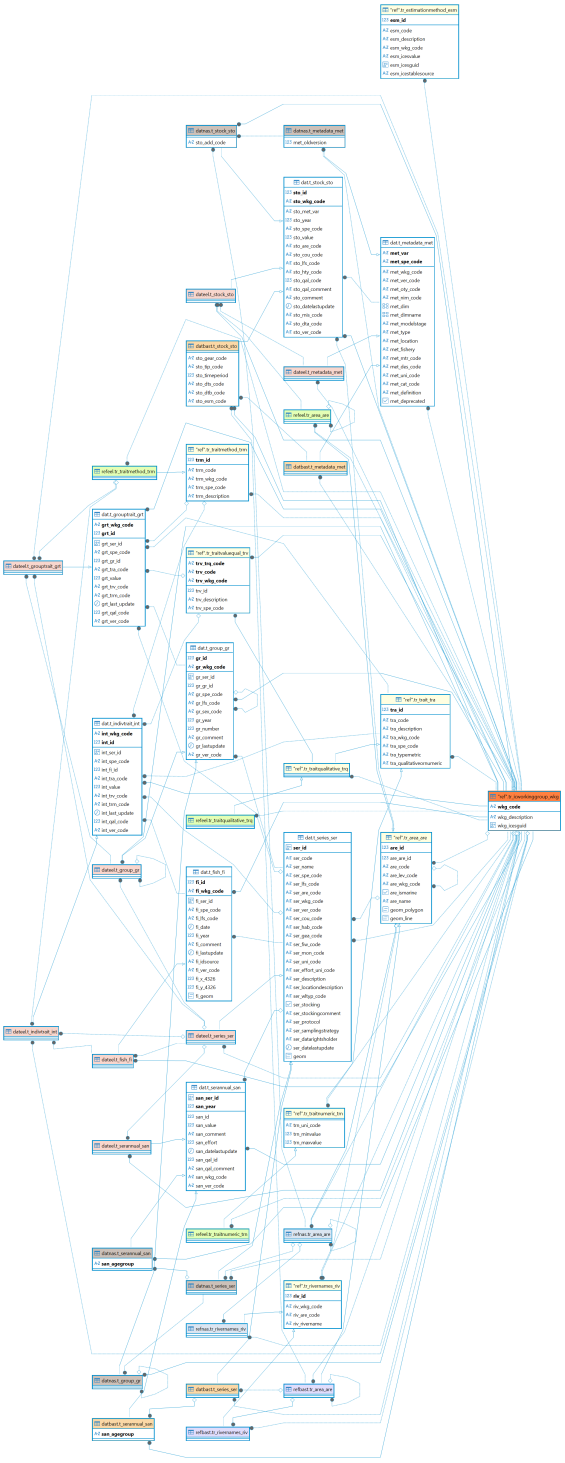


Figure 2.2: diagram of tr__icworkinggroup__wkg, click to enlarge

```
# RENAME TO "ref-countries-2024-01m-4326";


dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_country_cou ( cou_code,
    cou_country,
    cou_iso3code,
    geom,
    cou_order)
SELECT \"CNTR_ID\" AS cou_code, \"NAME_ENGL\" AS cou_country,  \"ISO3_CODE\"
AS cou_isocode, geom,
CASE WHEN \"CNTR_ID\" = 'GL' THEN 47
     WHEN \"CNTR_ID\" = 'CA' THEN 48
     ELSE 49 END AS cou_order
FROM  area.\"ref-countries-2024-01m-4326\"
WHERE \"CNTR_ID\" IN ('GL', 'CA', 'US');") #3

# Svalbard et Jan Mayen SJM NO Territory
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_country_cou ( cou_code,
    cou_country,
    cou_iso3code,
    geom,
    cou_order)
SELECT \"CNTR_ID\" AS cou_code, \"NAME_ENGL\" AS cou_country,  \"ISO3_CODE\"
AS cou_isocode, geom,
CASE WHEN \"CNTR_ID\" = 'GL' THEN 47
     WHEN \"CNTR_ID\" = 'CA' THEN 48
     ELSE 49 END AS cou_order
FROM  area.\"ref-countries-2024-01m-4326\"
WHERE \"CNTR_ID\" IN ('SJ');") #3

dbExecute(con_diaspara_admin,
          "UPDATE ref.tr_country_cou
SET geom = nuts.geom
FROM  area.\"ref-countries-2024-01m-4326\" nuts
WHERE nuts.\"CNTR_ID\" = tr_country_cou.cou_code;") # 40
```

```
tr_country_cou <- dbGetQuery(con_diaspara, "SELECT cou_code,cou_country,cou_order, cou_iso3c
knitr::kable(tr_country_cou) |> kable_styling(bootstrap_options = c("striped", "hover", "cor
```

Table 2.5: Country table in the diaspara DB

cou_code	cou_country	cou_order	cou_iso3code
IS	Iceland	0	ISL

NO	Norway	1	NOR
SE	Sweden	2	SWE
AX	Åland	3	ALA
FI	Finland	4	FIN
EE	Estonia	5	EST
LV	Latvia	6	LVA
LT	Lithuania	7	LTU
RU	Russia	8	RUS
PL	Poland	9	POL
CZ	Czech republic	10	CZE
DE	Germany	11	DEU
DK	Denmark	12	DNK
NL	Netherlands	13	NLD
BE	Belgium	14	BEL
LU	Luxembourg	15	LUX
IE	Ireland	16	IRL
GB	Great Britain	17	GBR
FR	France	18	FRA
ES	Spain	19	ESP
PT	Portugal	20	PRT
IT	Italy	21	ITA
MT	Malta	22	MLT
SI	Slovenia	23	SVN
HR	Croatia	24	HRV
BA	Bosnia-Herzegovina	25	BIH
ME	Montenegro	26	MNE
AL	Albania	27	ALB
GR	Greece	28	GRC
TR	Turkey	34	TUR
CY	Cyprus	35	CYP
SY	Syria	36	SYR
LB	Lebanon	37	LBN
IL	Israel	38	ISR
EG	Egypt	39	EGY
LY	Libya	40	LBY
TN	Tunisia	41	TUN
DZ	Algeria	42	DZA
MA	Morocco	43	MAR
VA	Vattican	46	VAT
GL	Greenland	47	GRL
CA	Canada	48	CAN
SJ	Svalbard and Jan Mayen	49	SJM

US	United States	49	USA

```
if (file.exists("data/country_sf.Rdata")) load("data/country_sf.Rdata") else {
  country_sf <- sf::st_read(con_diaspara,
                            query = "SELECT cou_code, ST_MakeValid(geom)
                           from ref.tr_country_cou") |>
    sf::st_transform(4326)
  save(country_sf, file="data/country_sf.Rdata")
}
#see here : https://stackoverflow.com/questions/70756215/
#plot-geodata-on-the-globe-perspective-in-r
# Note there is a problem of geometry for some of the polygons, and this require
# ST_Makevalid before intersection

# projection string used for the polygons & ocean background
crs_string <- "+proj=ortho +lon_0=-30 +lat_0=30"

# background for the globe - center buffered by earth radius
ocean <- sf::st_point(x = c(0,0)) |>
  sf::st_buffer(dist = 6371000) |>
  sf::st_sfc(crs = crs_string)
country_sf2 <-  country_sf |>
  sf::st_intersection(ocean |> sf::st_transform(4326)) |>
  # select visible area only
  sf::st_transform(crs = crs_string) # reproject to ortho
# now the action!
g <- ggplot(data = country_sf2) +
  geom_sf(data = ocean, fill = "aliceblue", color = NA) + # background first
  geom_sf(aes(fill = cou_code), lwd = .1) + # now land over the oceans
  scale_fill_discrete(guide = "none") +
  theme_void()

# this part is used to avoid long computations
png(filename="images/fig-country.png", bg="transparent")
print(g)
dev.off()
```

2.4 Unit (tr_units_uni)

SQL code to create tables

```
--DROP TABLE IF EXISTS ref.tr_units_uni CASCADE;

CREATE TABLE ref.tr_units_uni (
  uni_code varchar(20) NOT NULL,
```

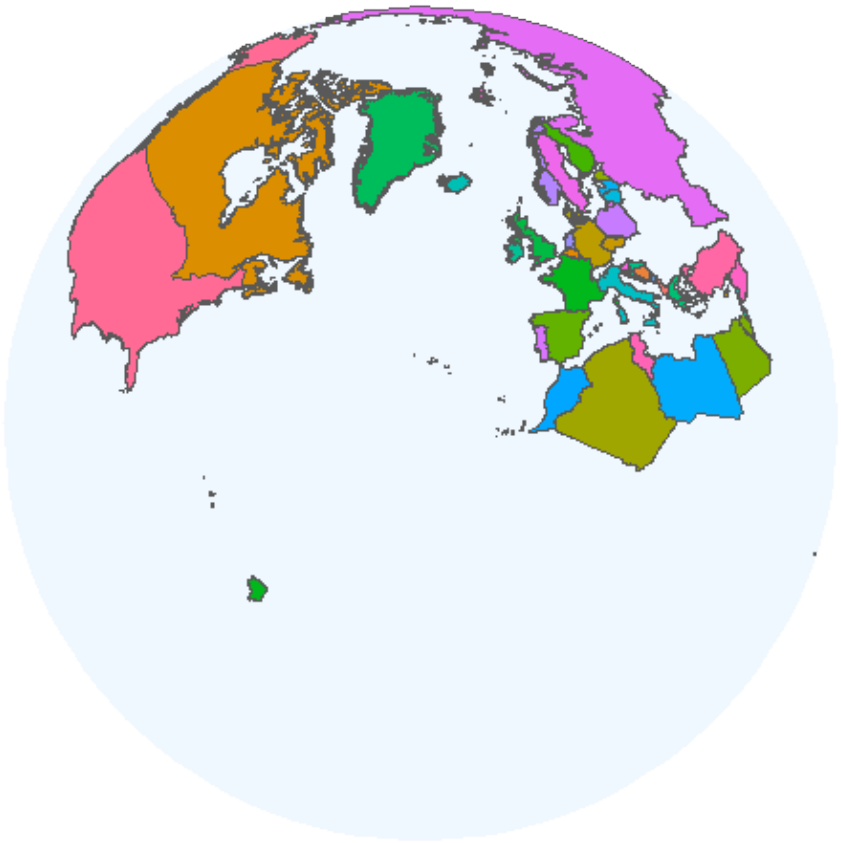


Figure 2.3: Map of countries in the diaspara DB © EuroGeographics

```
  uni_description text NOT NULL,
  uni_icesvalue character varying(4),
  uni_icesguid uuid,
  uni_icestablesource text,
  CONSTRAINT t_units_uni_pkey PRIMARY KEY (uni_code),
  CONSTRAINT uk_uni_description UNIQUE (uni_description),
  CONSTRAINT uk_uni_icesguid UNIQUE (uni_icesguid),
  CONSTRAINT uk_uni_icesvalue UNIQUE (uni_icesvalue)
);
GRANT ALL ON ref.tr_units_uni TO diaspara_admin;
GRANT SELECT ON ref.tr_units_uni TO diaspara_read;
-- I don't add definitions this is an ICES vocab
```

Creating the unit from wgeel and checking ICES code

First we import from wgeel

Then we standarize using ICES codes, it takes a while to scroll through the vocab. Sometimes several vocab are available for the same thing. We used the p06 as the most common source.

```
dbExecute(con_diaspara_admin,"INSERT INTO ref.tr_units_uni (
uni_code, uni_description)
SELECT * FROM refwgeel.tr_units_uni;")#25

dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='KGXX'
          where uni_code = 'kg';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='MTON'
          where uni_code = 't';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UCNT'
          where uni_code = 'nr';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UGRM'
          where uni_code = 'g';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UPMS'
          where uni_code = 'nr/m2';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UPMM'
          where uni_code = 'nr/m3';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UYRS'
          where uni_code = 'nr year';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UXMM'
          where uni_code = 'mm';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='NGPG'
          where uni_code = 'ng/g';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='HCTR'
          where uni_code = 'ha';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UTAA'
          where uni_code = 'nr day';")
```

```
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='NOPH'
          where uni_code = 'nr/h';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='NGPG'
          where uni_code = 'ng/g';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='UPCT'
          where uni_code = 'percent';")
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set
          (uni_icesvalue, uni_description)=
          ('XXXX', 'Not applicable (without unit)')
          where uni_code = 'wo';")

dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_units_uni
          VALUES ('year-1', 'Per year', 'XXPY');")
dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_units_uni
          VALUES ('s', 'Seconds', 'UTBB');")



p06 <- icesVocab::getCodeList('p06')
SamplingUnit <- icesVocab::getCodeList('SamplingUnit')
MUNIT <- icesVocab::getCodeList('MUNIT')
uni <- dbGetQuery(con_diaspara_admin, "SELECT * FROM ref.tr_units_uni;")
tempuni <- inner_join(uni, p06, by=join_by(uni_icesvalue==Key))
dbWriteTable(con_diaspara_admin, "tempuni", tempuni, overwrite=TRUE)
dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni
set uni_icesguid = \"GUID\"::uuid
          FROM tempuni
          where tempuni.uni_icesvalue=tr_units_uni.uni_icesvalue;") #16
dbExecute(con_diaspara_admin, "DROP TABLE tempuni;")

dbExecute(con_diaspara_admin,
          "UPDATE ref.tr_units_uni set uni_icestablesource = 'p06' where uni_icesvalue
IS NOT NULL AND
uni_icestablesource IS NULL;") # 16

query <- sprintf("INSERT INTO ref.tr_units_uni (uni_code,uni_description, uni_icesvalue, uni
                 "gd",
                 "Gear days for fyke/trap nets",
                 "gd",
                 "MUNIT",
                 "bf0570b7-45f2-41c7-9a46-de912a2b9ad4")
dbExecute(con_diaspara_admin,  query)


dbExecute(con_diaspara_admin, "UPDATE ref.tr_units_uni set uni_icesvalue='idx',
```

```
          uni_icestablesource = 'MUNIT',
          uni_icesguid ='87a9cf7f-fff4-4712-b693-76eec1403254'::uuid
          where uni_code = 'index';")

# p06[grep('Ton',p06$Description),c("Description","Key")]
# p06[grep('Without',tolower(p06$Description)),c("Description","Key")]
# p06[grep('nanogram',tolower(p06$Description)),c("Description","Key")]
# p06[grep('index',tolower(p06$Description)),c("Description","Key")]
# p06[grep('hour',tolower(p06$Description)),c("Description","Key")]
# p06[grep('kilogram',tolower(p06$Description)),c("Description","Key")]
# p06[grep('nanogram',tolower(p06$Description)),c("Description","Key")]
# p06[grep('haul',tolower(p06$Description)),c("Description","Key")]

dbExecute(con_diaspara_admin, "COMMENT ON TABLE ref.tr_units_uni IS
'Table of units, values from tables MUNIT and p06 have corresponding ICES code.'")
dbExecute(con_diaspara_admin, "COMMENT ON COLUMN ref.tr_units_uni.uni_code IS
'Unit code, lowercase, nr number, otherwise standard units.'")
dbExecute(con_diaspara_admin, "COMMENT ON COLUMN ref.tr_units_uni.uni_description
 IS 'Unit code, lowercase, nr number, otherwise standard units.'")
dbExecute(con_diaspara_admin, "COMMENT ON COLUMN ref.tr_units_uni.uni_icesvalue IS
'ICES code standard from the British Oceanographic Data Centre (p06) or MUNIT
table.';")
dbExecute(con_diaspara_admin,
          "COMMENT ON COLUMN ref.tr_units_uni.uni_icestablesource IS
'Table source in ICES.';")
dbExecute(con_diaspara_admin,
          "COMMENT ON COLUMN ref.tr_units_uni.uni_icesguid IS
'GUID, type https://vocab.ices.dk/?codetypeguid=<guidcode> to get access to the
vocab in ICES.';")
dbExecute(con_diaspara_admin, "GRANT ALL ON TABLE ref.tr_units_uni
          to diaspara_admin;")
dbExecute(con_diaspara_admin, "GRANT SELECT ON TABLE ref.tr_units_uni
          to diaspara_read;")
#for WGBAST
#
query <- sprintf("INSERT INTO ref.tr_units_uni (uni_code,uni_description, uni_icesvalue, uni
                 "nd",
                 "Net-days (fisheries)",
                 "nd",
                 "MUNIT",
                 "f2783f1c-defa-4551-a9e3-1cfa173a0b9f")
dbExecute(con_diaspara_admin,  query)
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_units_uni;")|>
knitr::kable() |>
```

```
kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.6: tr_units_uni table, check missing values currently
Vocab

uni_code	uni_description	uni_icesv
kg/d	kilogramme per day	NA
kg/boat/d	kilogramme per boat per day	NA
nr haul	number of haul	NA
nr electrofishing	number of electrofishing campain in the year to collect the recruitment index	NA
nr/haul	number per haul	NA
kg/ha	weight in kilogrammes per surface in hectare	NA
nr net.night	number of net and night	NA
nr fyke.day	number of fyke and day	NA
nr site	number of site	NA
nr/net/day	number per net and day	NA
kg	weight in kilogrammes	KGXX
t	weight in tonnes	MTON
nr	number	UCNT
g	gram	UGRM
nr/m2	number per square meter	UPMS
nr/m3	number per cubic meter	UPMM
nr year	number of years	UYRS
mm	milimeter	UXMM
ha	Surface	HCTR
nr day	number of days	UTAA
nr/h	number per hour	NOPH
ng/g	nanogram per gram	NGPG
percent	percentage	UPCT
wo	Not applicable (without unit)	XXXX
year-1	Per year	XXPY
s	Seconds	UTBB
gd	Gear days for fyke/trap nets	gd
index	calculated value following a specified protocol	idx
nd	Net-days (fisheries)	nd

Some of the values are missing from ICES vocab (Table Table 2.6).

QUESTION ICES: missing values for units, what do we do ?

What do we do with units without correspondance? These come from FAO (if I remember well). Do we try to search for those in the wgeel database, and then remove if not existing or try to change existing values ?

- Kg/day there is a kg/hour do we need to change to that type and convert existing series ?
- Nr haul There is a definition of haul in the ICES vocab but it seems very related to sampling box, basket. And it's not the number of haul.
- Before working any further I would like your opinion there.

Answer ICES

WGEEL: New MUNIT codes #737

2.5 Parameters

Parameters are a simple way to reduce the complexity of data. It will correspond to all nimble variables, reduced to their lower level (e.g. 3 dimensional arrays with dimensions [area, year, stage] will be translated as many lines with the corresponding values in columns area, year, and stage), and the identifyer of the variable will be used for all the lines necessary to store this dataset. In practise, parameters also correspond to input data, and output data in the model. The parameters will be described by their metadata as illustrated in Figure 2.5

We can have a look at the metadata in the analysis done on the WGNAS database WGNAS description.

There is a problem with some columns with mutiple dimensions which need to be aligned. For instance a column can hold year, or age. This will be solved by using an additional column where data can be of several types (see tg_additional_add in Section 4.4). The description could be used within a type array. SQL server does not work with array so it's not a good idea to use those.

Checking stock codes using icesASD and icesSD packages

> DIASPARA
> Environment (sea, transition …), age, life stage and complex are in the metadata. But they are also in the main table. The `complex` will be derived from the spatial structure still in construction

> DIASPARA
> As in the diagram, we added a category (data type in Figure 2.5). The idea is to be able to get quickly all parameters related to a type, e.g. catch, mortality, biomass.

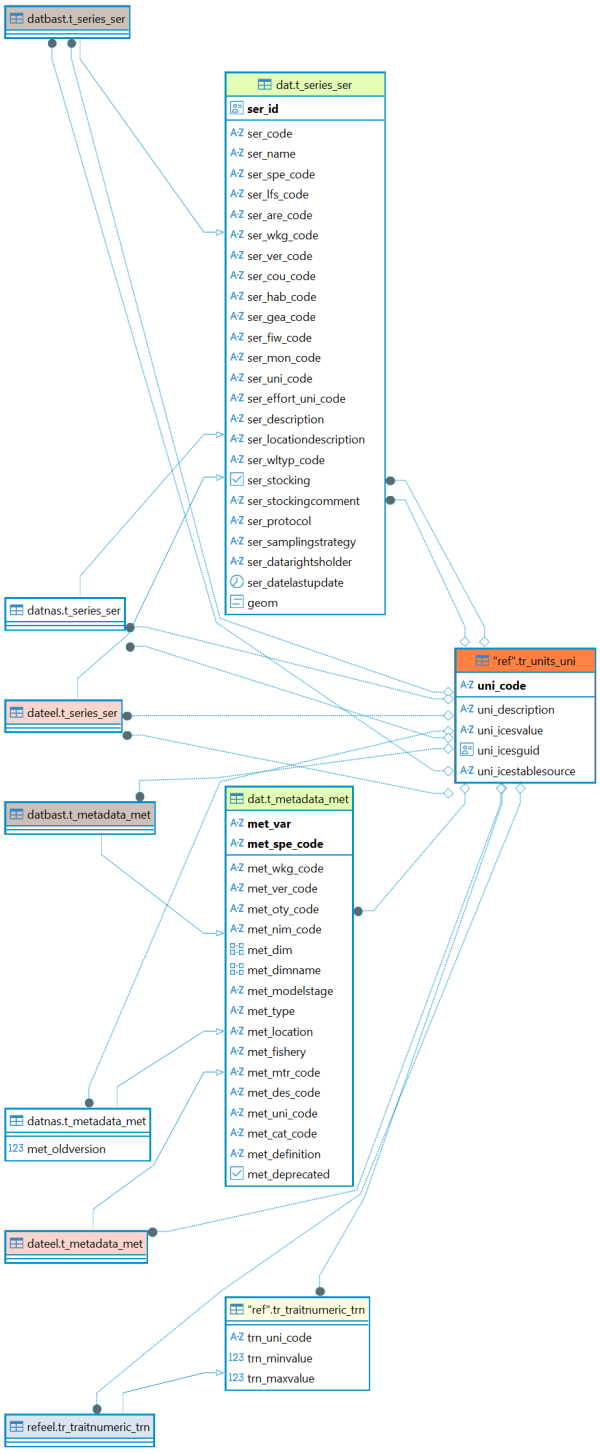


Figure 2.4: diagram of tr_units_uni

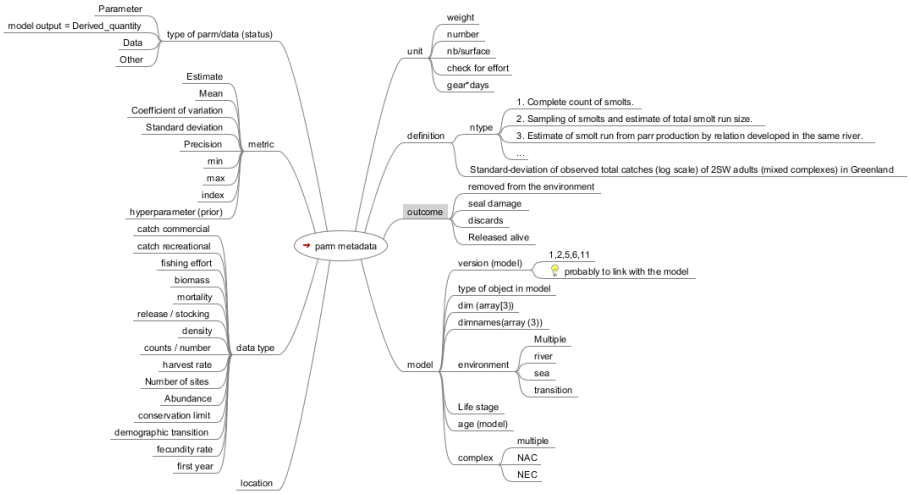


Figure 2.5: Mind map of the metadata structure

Table 2.7: Access to the advice using icesAdvice

```
# install.packages('icesAdvice', repos = c('https://ices-tools-prod.r-universe.dev', 'https
# install.packages("icesSD", repos = c("https://ices-tools-prod.r-universe.dev", "https://cl

library('icesAdvice')
library('icesSAG')
library('icesSD')
# this does not give the
advice <- getAdviceViewRecord()
advice[grepl('ele',advice$stockCode),
       c('adviceDOI', 'stockCode','assessmentyear')] |> kable
sd <- mapply(getSD, year= 2020:2024, SIMPLIFY=FALSE)
sd <- do.call(rbind,sd)
ww <- grepl('ele',sd$StockKeyLabel) | grepl('Salmo',sd$speciesScientificName)
sd[ww,] |> kable()
```

DIASPARA
WGBAST, WGNAS, WGEEL, WGTRUTTA will have to check definitions in tr_destination_dest. Note this is just a column in t_metadata_met not in the main table.

2.6 Object type (tr_objectype_oty)

This table (Table 2.8) is used in metadata

SQL code to create tables

```
DROP TABLE IF EXISTS ref.tr_objecttype_oty CASCADE;
CREATE TABLE ref.tr_objecttype_oty (
oty_code TEXT PRIMARY KEY,
oty_description TEXT
);

INSERT INTO ref.tr_objecttype_oty VALUES ('Single_value', 'Single value');
INSERT INTO ref.tr_objecttype_oty VALUES ('Vector', 'One dimension vector');
INSERT INTO ref.tr_objecttype_oty VALUES ('Matrix', 'Two dimensions matrix');
INSERT INTO ref.tr_objecttype_oty VALUES ('Array', 'Three dimensions array');

COMMENT ON TABLE ref.tr_objecttype_oty IS
'Table indicating the dimensions of the object stored in the model,
single value, vector, matrix, array';

COMMENT ON COLUMN ref.tr_objecttype_oty.oty_code IS
'code of the object type, single_value, vector, ...';

COMMENT ON COLUMN ref.tr_objecttype_oty.oty_code IS 'description of the object type';
GRANT ALL ON ref.tr_objecttype_oty TO diaspara_admin;
GRANT SELECT ON ref.tr_objecttype_oty TO diaspara_read;
```

Table 2.8: Object type

oty_code	oty_description
Single_value	Single value
Vector	One dimension vector
Matrix	Two dimensions matrix
Array	Three dimensions array

2.7 Type of parm / data (tr_nimble_nim)

In the salmoglob database (WGNAS), this table (Table 2.9) corresponds to both tables `status` and `nimble` which most often contain the same information.

SQL code to create tables

```
--nimble

DROP TABLE IF EXISTS ref.tr_nimble_nim CASCADE;
CREATE TABLE ref.tr_nimble_nim (
nim_code TEXT PRIMARY KEY,
nim_description TEXT
);

COMMENT ON TABLE ref.tr_nimble_nim IS
'Indicate the type of data, parameter constant, parameter estimate, output, other ...';
-- Note this is a mix of nimble and status, which mean the same....

INSERT INTO ref.tr_nimble_nim VALUES ('Data', 'Data entry to the model');
INSERT INTO ref.tr_nimble_nim
VALUES ('Parameter constant', 'Parameter input to the model');
INSERT INTO ref.tr_nimble_nim
VALUES ('Parameter estimate', 'Parameter input to the model');
INSERT INTO ref.tr_nimble_nim
VALUES ('Output', 'Output from the model, derived quantity');
-- Do we want another type here ?
--INSERT INTO ref.tr_nimble_nim VALUES ('observation', 'Observation not used in the model');
INSERT INTO ref.tr_nimble_nim
VALUES ('Other', 'Applies currently to conservation limits');
GRANT ALL ON ref.tr_nimble_nim TO diaspara_admin;
GRANT SELECT ON ref.tr_nimble_nim TO diaspara_read;
```

Table 2.9: Nimble

nim_code	nim_description
Data	Data entry to the model
Parameter constant	Parameter input to the model
Parameter estimate	Parameter input to the model
Output	Output from the model, derived quantity
Other	Applies currently to conservation limits

2.8 Version (tr_version_ver)

Currently in the salmoglob information about version only correspond to different versions of the same parameter see WGNAS analysis report. The `database` only contains the variables used to run the salmoglob model, at the time of the working group, any updated value is copied to the `database_archive` which then contains historical values. From this it might be possible to rebuild historical states of the database but it is not straightforward, as the archive database can hold, for the same year, multiple values of the same variable, if the corrections or updates were made several times. The dates (column date_time) used in the `database` and `database_archive` give information about the latest update. An analysis with Pierre Yves Hernwann on unique values for tupples [version, year, type, age, area, location, metric, var_mod] shows that some duplicates are present in the archive database in 2021 and 2022, so by using the latest date in the year it is possible to reproduce the state of the database at the moment of the working group. Still it was agreed that a clear versioning would ease up the work (as in WGEEL). It was also agreed that metadata should also contain information about historical variables For instance we create a new variable, so we know when it was introduced. So the version column was added to we added to both metadata (`t_metadata_met`) table and stock table (`t_stock_sto`). Some variables might get deprecated over time.

> DIASPARA
> Note that the version (Table 2.10) contains reference to the datacall. The version will be handled in an inherited table for WGNAS, WGEEL and WGBAST.

Question to WNGAS

Can you explain the versions

Answer WGNAS (Pierre-Yves Hernvann)

The version number are related to variables, each time they are changed a new variable number is created in metadata by the shiny and the date is set, so we can keep track of the variables. There is also an information on who did the change (not accessible to the public). Pierre Yves agrees that saving the database at each working group is probably the best way to re-run the model at that stage.

> TODO ICES
> The DB will be held in ICES server. We will need a procedure to save the database at each working group to be able to run past versions of the model. This means keeping a database_archive for each working groups. It's a straightforward copy of the `t_stock_sto` for each working group.

SQL code to create tables

```
-- It seems to me that metadata should contain information about historical
-- variables, so I'm moving this from the main table and adding to metadata
-- some variables might get deprecated in time.
-- Unless they get a new version this might not change
-- I have removed reference to stockkey as there are several stock keys
-- for the work of WGNAS
DROP TABLE IF EXISTS ref.tr_version_ver CASCADE;
CREATE TABLE ref.tr_version_ver(
ver_code TEXT PRIMARY KEY,
ver_year INTEGER NOT NULL,
ver_spe_code CHARACTER VARYING(3),
/*CONSTRAINT fk_ver_spe_code FOREIGN KEY (ver_spe_code)
REFERENCES ref.tr_species_spe(spe_code)
ON UPDATE CASCADE ON DELETE RESTRICT,*/
ver_wkg_code TEXT NOT NULL,
CONSTRAINT fk_ver_wkg_code  FOREIGN KEY (ver_wkg_code)
REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
ON UPDATE CASCADE ON DELETE RESTRICT,
--ver_stockkey INTEGER NOT NULL,
ver_stockkeylabel TEXT,
---ver_stockadvicedoi TEXT NOT NULL,
ver_datacalldoi TEXT NULL,
ver_version INTEGER NOT NULL,
ver_description TEXT
);
COMMENT ON TABLE ref.tr_version_ver
IS 'Table of data or variable version, essentially one datacall or advice.';
COMMENT ON COLUMN ref.tr_version_ver.ver_code
IS 'Version code, stockkey-year-version.';
COMMENT ON COLUMN ref.tr_version_ver.ver_year
IS 'Year of assessement.';
COMMENT ON COLUMN ref.tr_version_ver.ver_spe_code
IS 'Species code e.g. ''Anguilla anguilla'' or ''Salmo salar,Salmo trutta'' the reference na
COMMENT ON COLUMN ref.tr_version_ver.ver_wkg_code
IS 'Code of the working group, one of WGBAST, WGEEL, WGNAS, WKTRUTTA';
--COMMENT ON COLUMN ref.tr_version_ver.ver_stockkey
--IS 'Stockkey (integer) from the stock database.';
COMMENT ON COLUMN ref.tr_version_ver.ver_stockkeylabel
IS 'Ver_stockkeylabel e.g. ele.2737.nea.';
--COMMENT ON COLUMN ref.tr_version_ver.ver_stockadvicedoi
--IS 'Advice DOI corresponding to column adviceDOI
--when using icesASD::getAdviceViewRecord().';
COMMENT ON COLUMN ref.tr_version_ver.ver_datacalldoi
IS 'Data call DOI, find a way to retreive that information
```

```
and update this comment';
COMMENT ON COLUMN ref.tr_version_ver.ver_version
IS 'Version code in original database, eg 2,4 for wgnas, dc_2020 for wgeel.';
COMMENT ON COLUMN ref.tr_version_ver.ver_description
IS 'Description of the data call / version.';

GRANT ALL ON ref.tr_version_ver TO diaspara_admin;
GRANT SELECT ON ref.tr_version_ver TO diaspara_read;


-- we need to allow several species separated by a comma (several species for a working grou
-- the
ALTER TABLE ref.tr_version_ver DROP CONSTRAINT fk_ver_spe_code;
ALTER TABLE refnas.tr_version_ver DROP CONSTRAINT fk_ver_spe_code;
ALTER TABLE refeel.tr_version_ver DROP CONSTRAINT fk_ver_spe_code;
ALTER TABLE refbast.tr_version_ver DROP CONSTRAINT fk_ver_spe_code;
ALTER TABLE ref.tr_version_ver ALTER COLUMN ver_spe_code type TEXT;
```

```
#sd <-do.call(rbind,mapply(icesSD::getSD, year= 2020:2024, SIMPLIFY=FALSE))
#sd[grepl('Working Group on North Atlantic Salmon',sd$ExpertGroupDescription),]



tr_version_ver <- data.frame(
  ver_code = paste0("WGNAS-",2020:2024,"-1"),
  ver_year = 2020:2024,
  ver_spe_code = "127186",
  ver_wkg_code = "WGNAS",
  ver_datacalldoi=c(NA,NA,NA,NA,"https://doi.org/10.17895/ices.pub.25071005.v3"),
  ver_stockkeylabel =c("sal.neac.all"), # sugested by Hilaire.
  # TODO FIND other DOI (mail sent to ICES)
  ver_version=c(1,1,1,1,1), # TODO WGNAS check that there is just one version per year
  ver_description=c(NA,NA,NA,NA,NA)) # TODO WGNAS provide model description

DBI::dbWriteTable(con_diaspara_admin, "temp_tr_version_ver", tr_version_ver,
                  overwrite = TRUE)
dbExecute(con_diaspara_admin, "INSERT INTO refnas.tr_version_ver(ver_code, ver_year, ver_spe
DBI::dbExecute(con_diaspara_admin, "DROP TABLE temp_tr_version_ver;")
```

ver_code	ver_year	ver_spe_code	ver_stockkeylabel	ver_datacalldoi
WGNAS-2020-1	2020	Salmo salar	sal.neac.all	

WGNAS-2021-1	2021	Salmo salar	sal.neac.all	
WGNAS-2022-1	2022	Salmo salar	sal.neac.all	
WGNAS-2023-1	2023	Salmo salar	sal.neac.all	
WGNAS-2024-1	2024	Salmo salar	sal.neac.all	
DIASPARA-2025-1	2025	Salmo salar	NA	NA
WGEEL-2023-1	2023	Anguilla anguilla	ele	
WGEEL-2024-1	2024	Anguilla anguilla	ele	https://doi.org/10.17895/
WGEEL-2025-1	2025	Anguilla anguilla	ele	https://doi.org/10.17895/
WGEEL-2025-2	2025	Anguilla anguilla	ele	https://doi.org/10.17895/
WGEEL-2016-1	2016	Anguilla anguilla	ele	
WGEEL-2017-2	2017	Anguilla anguilla	ele	
WGEEL-2017-1	2017	Anguilla anguilla	ele	
WGEEL-2018-1	2018	Anguilla anguilla	ele	
WGEEL-2019-1	2019	Anguilla anguilla	ele	
WGEEL-2020-1	2020	Anguilla anguilla	ele	
WGEEL-2021-1	2021	Anguilla anguilla	ele	
WGEEL-2022-1	2022	Anguilla anguilla	ele	
WGBAST-2024-1	2024	Salmo salar, Salmo trutta	sal.27.22–31	https://doi.org/10.17895/
WGBAST-2025-1	2025	Salmo salar, Salmo trutta	sal.27.22–31	https://doi.org/10.17895/

QUESTION to ICES: What is the vocabulary for datacalls ?

We would like to access to this table : datacall (see link in ICES webpage). Currently we see the current year, this is nice, how do we access to historical data, is there a way to get it using a query ? We've found a link for advice or stocks but not data calls.

2.9 Metric (tr_metric_mtr)

SQL code to create tables

```
-- metric

DROP TABLE IF EXISTS  ref.tr_metric_mtr CASCADE;
CREATE TABLE ref.tr_metric_mtr(
mtr_code TEXT PRIMARY KEY,
mtr_description TEXT
);


INSERT INTO ref.tr_metric_mtr VALUES
('Estimate' , 'Estimate');
INSERT INTO ref.tr_metric_mtr VALUES
('Index', 'Index');
```

```
INSERT INTO ref.tr_metric_mtr VALUES
('Bound', 'Either min or max');
INSERT INTO ref.tr_metric_mtr VALUES
('Hyperparameter', 'Hyperparameter (prior)');
INSERT INTO ref.tr_metric_mtr VALUES
('SD', 'Standard deviation');
INSERT INTO ref.tr_metric_mtr VALUES
('CV', 'Coefficient of variation');
INSERT INTO ref.tr_metric_mtr VALUES
('Precision', 'Inverse of variance');
INSERT INTO ref.tr_metric_mtr VALUES
('Mean', 'Mean');
INSERT INTO ref.tr_metric_mtr VALUES
('Min','Minimum');
INSERT INTO ref.tr_metric_mtr VALUES
('Max','Maximum');

GRANT ALL ON ref.tr_metric_mtr TO diaspara_admin;
GRANT SELECT ON ref.tr_metric_mtr TO diaspara_read;
COMMENT ON TABLE ref.tr_metric_mtr IS
'Table metric describe the type of parm used, Index, Bound ...';
```

Table 2.11: Metric, type of parm used in the model

mtr_code	mtr_description
Estimate	Estimate
Index	Index
Bound	Either min or max
Hyperparameter	Hyperparameter (prior)
SD	Standard deviation
CV	Coefficient of variation
Precision	Inverse of variance
Mean	Mean
Min	Minimum
Max	Maximum

NOTE
This list currenly correspond to the needs of both WGNAS and WGBAST. But the metric can be NULL, for instance in case of a number of fish released, none of the above (Table 2.11) would apply.

2.10 Category (tr_category_cat)

Categories Table 2.12 were in the salmoglob metadata, here they were simplified to be able to get groups of parameters, for instance all parameters dealing with catch.

SQL code to create tables

```
-- tr_category_cat

DROP TABLE IF EXISTS ref.tr_category_cat CASCADE;
CREATE TABLE ref.tr_category_cat (
cat_code TEXT PRIMARY KEY,
cat_description TEXT
);

INSERT INTO ref.tr_category_cat VALUES
('Catch', 'Catch, including recreational and commercial catch.');
INSERT INTO ref.tr_category_cat VALUES (
'Effort', 'Parameter measuring fishing effort.');
INSERT INTO ref.tr_category_cat VALUES (
'Biomass', 'Biomass of fish either in number or weight.');
INSERT INTO ref.tr_category_cat VALUES (
'Mortality', 'Mortality either expressed in year-1 (instantaneous rate)
as F in exp(-FY) but can also be harvest rate.');
INSERT INTO ref.tr_category_cat VALUES (
'Release', 'Release or restocking.');
INSERT INTO ref.tr_category_cat VALUES (
'Density', 'Fish density.');
INSERT INTO ref.tr_category_cat VALUES (
'Count', 'Count or abundance or number of fish.');
INSERT INTO ref.tr_category_cat VALUES (
'Conservation limit', 'Limit of conservation in Number or Number of eggs.');
INSERT INTO ref.tr_category_cat VALUES (
'Life trait', 'Life trait parameterized in model, e.g. growth parameter,
fecundity rate ...');
INSERT INTO ref.tr_category_cat VALUES (
'Other', 'Other variable/ parameter used in the model other than the previous categories,
origin distribution of catches, proportions, parameters setting the beginning and ending dat
COMMENT ON TABLE ref.tr_category_cat IS
'Broad category of data or parameter, catch, effort, biomass, mortality, count ...,
 more details in the table ref.tr_parameter_parm e.g. commercial catch,
recreational catch are found in the parameter value and definition and unit,
this list is intended to be short.';

GRANT ALL ON ref.tr_category_cat TO diaspara_admin;
GRANT SELECT ON ref.tr_category_cat TO diaspara_read;
```

Table 2.12: category of par

cat_code	cat_description
Catch	Catch, including recreational and commercial catch.
Effort	Parameter measuring fishing effort.
Biomass	Biomass of fish either in number or weight.
Mortality	Mortality either expressed in year-1 (instantaneous rate) as F in exp(-FY) but can also
Release	Release or restocking.
Density	Fish density.
Count	Count or abundance or number of fish.
Conservation limit	Limit of conservation in Number or Number of eggs.
Life trait	Life trait parameterized in model, e.g. growth parameter, fecundity rate ...
Other	Other variable/ parameter used in the model other than the previous categories, origin

2.11 Destination (tr_destination_dest)

This table was added for WGBAST. The idea is "what becomes of this fish". It allows to integrate discards, releases and seal damages.

SQL code to create tables

```
-- table ref.tr_destination_des

DROP TABLE IF EXISTS ref.tr_destination_des CASCADE;
CREATE TABLE ref.tr_destination_des (
des_code TEXT PRIMARY KEY,
des_description TEXT
);

COMMENT ON TABLE ref.tr_destination_des IS
'Table of fish destination. When dealing with fish, e.g. in landings,what is the future of t
Removed (from the environment)';
INSERT INTO ref.tr_destination_des VALUES
('Removed', 'Removed from the environment, e.g. caught and kept');
INSERT INTO ref.tr_destination_des VALUES (
'Seal damaged', 'Seal damage');
INSERT INTO ref.tr_destination_des VALUES (
'Discarded', 'Discards');
INSERT INTO ref.tr_destination_des VALUES (
'Released', 'Released alive');
INSERT INTO ref.tr_destination_des VALUES (
'Released ', 'Released alive');
```

```
GRANT ALL ON ref.tr_destination_des TO diaspara_admin;
GRANT SELECT ON ref.tr_destination_des TO diaspara_read;
```

Table 2.13: category of parameters

des_code	des_description
Removed	Removed from the environment, e.g. caught and kept
Seal damaged	Seal damage
Discarded	Discards
Released	Released alive

> NOTE DIASPARA
> Hilaire noted that naming the table "outcome" wasn't ideal so we've followed his suggestion

2.12 Area (tr_area_are)

This table has been created here but see the habitat report for the full habitat referential creation.

2.12.1 Habitat level (tr_habitatlevel_lev)

SQL code to create tables

```
--DROP TABLE IF EXISTS ref.tr_habitatlevel_lev CASCADE;

CREATE TABLE ref.tr_habitatlevel_lev(
   lev_code TEXT PRIMARY KEY,
   lev_description TEXT
);

COMMENT ON TABLE ref.tr_habitatlevel_lev
IS 'Table of geographic levels stock, complex, country, region, basin, river,
the specific order depend according to working groups.';

GRANT ALL ON ref.tr_habitatlevel_lev TO diaspara_admin;
GRANT SELECT ON ref.tr_habitatlevel_lev TO diaspara_read;
```

```
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Panpopulation',
  'This is the highest geographic level for assessement.'
  );")
```

```
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Complex',
  'Corresponds to large sublevels at which the Panpopulation is assessed, e.g.
  NAC NEC for WGNAST, Gulf of Bothnia for WGBAST, Mediterranean for WGEEL.'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Stock',
  'Correspond to stock units for which advices are provided in ICES, this can be the level c
  or another level e.g. .'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Country',
  'Corresponds to one or more units, but in almost all stocks
  this level is relevant to split data.'
  );")


dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'EMU',
  'Administrative unit for eel, the hierarchical next level is country.'
  );")

# note this can be unit or Asssessment unit it can have two meanings ...
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Assessment_unit',
  'Corresponds to an assessment unit in the Baltic sea, and area for
  WGNAS, and EMU for WGEEL.'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Regional',
  'Corresponds to subunits of stock assessment units or
  basins grouping several river. Although it is not used yet for
  some models, regional genetic difference or difference in stock
  dynamic support collecting a regional level.'
  );")
```

```
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'River',
  'One river is a unit corresponding practically almost always to a watershed.'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'River_section',
  'Section of river, only a part of a basin, for instance to separate between
  wild and mixed river category in the Baltic.'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Major',
  'Major fishing areas from ICES.'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Subarea',
  'Subarea from ICES, FAO and NAFO'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Division',
  'Division from ICES, GFCM and NAFO'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Subdivision',
  'Subdivision level from ICES, GFCM and NAFO'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
  'Lagoons',
  'Shallow body of water seperated from a larger body of water by a narrow landform'
  );")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_habitatlevel_lev VALUES(
```

```
  'Subdivision_grouping',
  'Groups of subdivision from ICES used in the Baltic'
  );")

dbExecute(con_diaspara_admin,"UPDATE ref.tr_habitatlevel_lev
    SET lev_description='Corresponds to an assessment unit in the Baltic sea, and area for W
    WHERE lev_code='Assessment_unit'"); # remove spaces ...
```

Table 2.14:

lev_code	lev_description
Stock	This is the highest geographic level for assessement, stock level.
Complex	Corresponds to large sublevels at which the stock is assessed, e.g. NAC NEC for W
Country	Corresponds to one or more units, but in almost all stocks this level is relevant to s
Assessment_unit	Corresponds to an assessment unit in the Baltic sea, and area for WGNAS, and EM
Regional	Corresponds to subunits of stock assessment units or basins grouping several river.
River	One river is a unit corresponding practically almost always to a watershed.
Major	Major fishing areas from ICES.
Subarea	Subarea from ICES, FAO and NAFO
Division	Division from ICES, GFCM and NAFO
Subdivision	Subdivision level from ICES, GFCM and NAFO
Fisheries	Specific fisheries area used by some working groups (WGNAS), e.g. FAR fishery, G
EMU	Administrative unit for eel, the hierarchical next level is country.
River_section	Section of river, only a part of a basin, for instance to separate between wild and m
Subdivision_grouping	Groups of subdivision from ICES used in the Baltic
Lagoons	Shallow body of water seperated from a larger body of water by a narrow landform

2.12.2 Area (tr_area_are)

Again, this table has been created in the habitat report.

SQL code to create tables

```
--DROP TABLE IF EXISTS ref.tr_area_are CASCADE;
CREATE TABLE ref.tr_area_are (
   are_id INTEGER PRIMARY KEY,
   are_are_id INTEGER,
   are_code  TEXT,
   are_lev_code TEXT,
   are_wkg_code TEXT,
   are_ismarine BOOLEAN,
   are_name TEXT,
   geom_polygon geometry(MULTIPOLYGON, 4326),
   geom_line geometry(MULTILINESTRING, 4326),
```

```
  CONSTRAINT fk_are_are_id FOREIGN KEY (are_are_id)
  REFERENCES ref.tr_area_are (are_id) ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT uk_are_code UNIQUE (are_code),
  CONSTRAINT fk_area_lev_code FOREIGN KEY (are_lev_code) REFERENCES
  ref.tr_habitatlevel_lev(lev_code) ON UPDATE CASCADE ON DELETE CASCADE,
  CONSTRAINT fk_area_wkg_code FOREIGN KEY (are_wkg_code) REFERENCES
  ref.tr_icworkinggroup_wkg(wkg_code) ON UPDATE CASCADE ON DELETE CASCADE
);

GRANT ALL ON ref.tr_area_are TO diaspara_admin;
GRANT SELECT ON ref.tr_area_are TO diaspara_read;

COMMENT ON TABLE ref.tr_area_are IS 'Table corresponding to different geographic levels, fro
to river section.');

-- we need to rename the column (fix 16/10/2025)
ALTER TABLE "ref".tr_area_are RENAME COLUMN are_rivername TO are_name;
```

Table 2.15: Geographic areas

are_id	are_are_id	are_code	are_lev_code	are_wkg_code	are_ismarine
306	17	2120027350	River	WGBAST	FALSE
3099	174	20127557	River_section	WGBAST	FALSE
3086	174	20127414	River_section	WGBAST	FALSE
3088	174	20127330	River_section	WGBAST	FALSE
15	3	3 Bothnian Sea	Assessment_unit	WGBAST	FALSE
14	3	2 Western Bothnian Bay	Assessment_unit	WGBAST	FALSE
17	3	5 Eastern Main Basin	Assessment_unit	WGBAST	FALSE
18	3	6 Gulf of Finland	Assessment_unit	WGBAST	FALSE
13	3	1 Northeastern Bothnian Bay	Assessment_unit	WGBAST	FALSE
16	3	4 Western Main Basin	Assessment_unit	WGBAST	FALSE

NOTE DIASPARA
Areas are specific to each working group (see Figure 2.10)

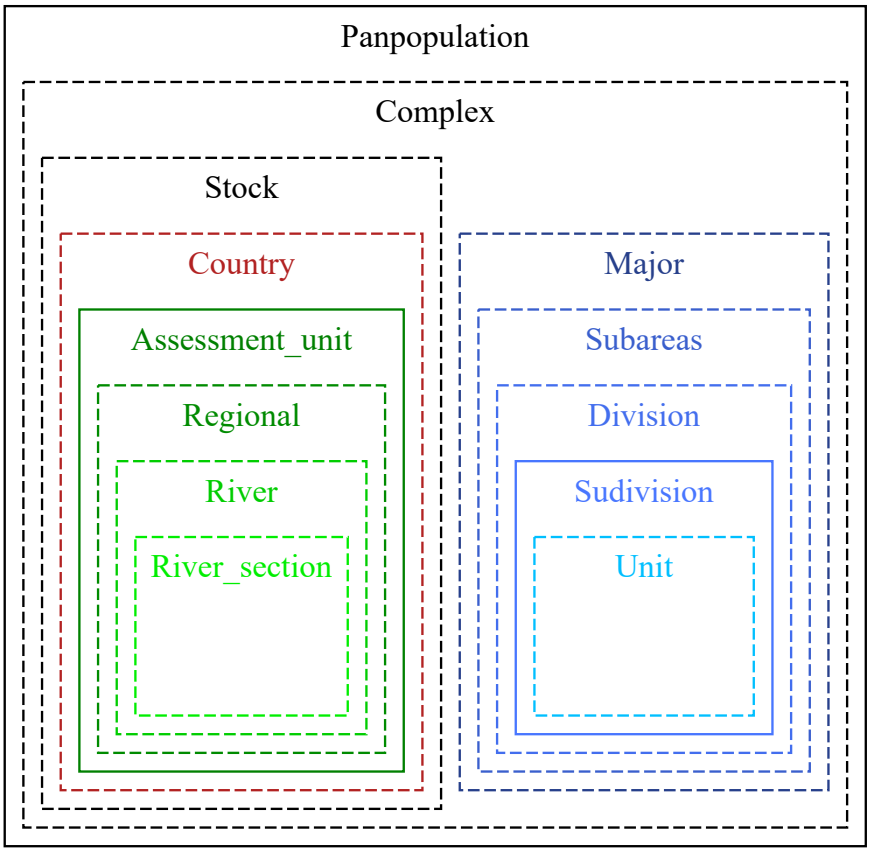


Figure 2.6: First concepts of the hierarchy

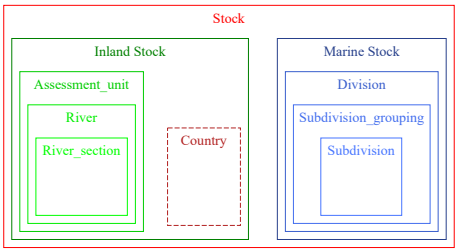


Figure 2.7: WGBAST

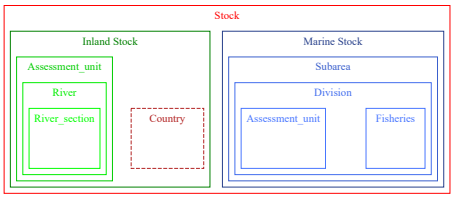


Figure 2.8: WGNAS

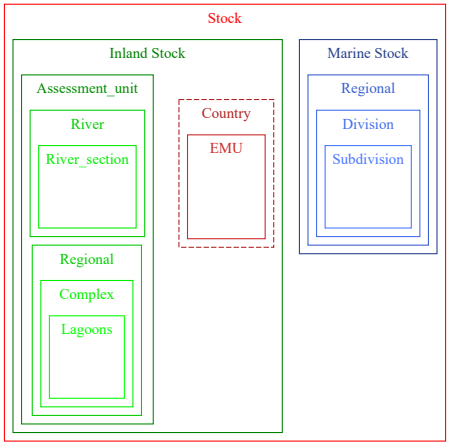


Figure 2.9: WGEEL

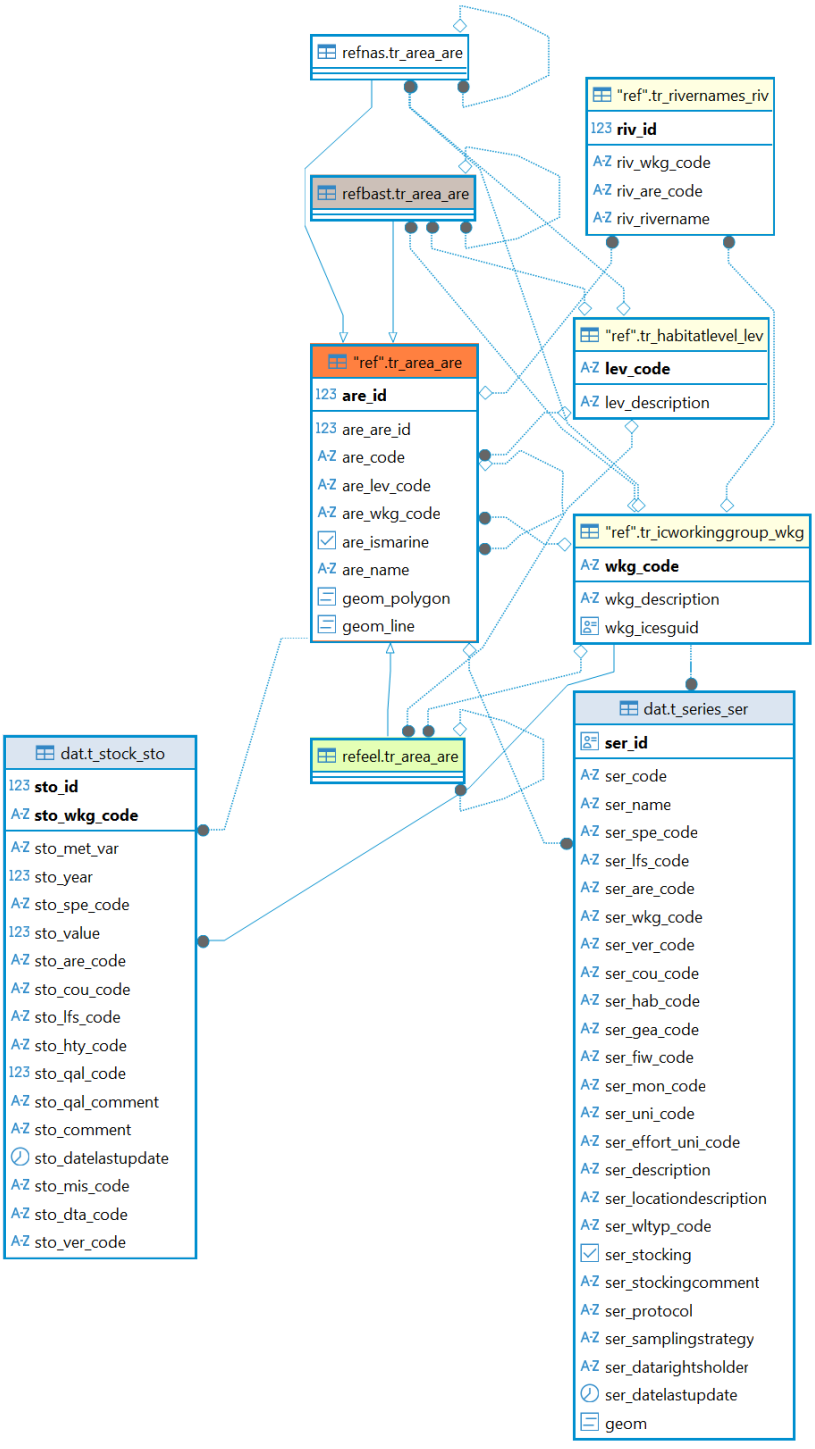


Figure 2.10: diagram of tr_area_are

2.13 Data access (tr_dataaccess_dta)

Type of data Public, or Restricted

SQL code to create tables

```
--DROP TABLE IF EXISTS ref.tr_dataaccess_dta CASCADE;
CREATE TABLE ref.tr_dataaccess_dta(
   dta_code TEXT PRIMARY KEY,
   dta_description TEXT
);
GRANT ALL ON ref.tr_dataaccess_dta  TO diaspara_admin;
GRANT SELECT ON ref.tr_dataaccess_dta TO diaspara_read;
COMMENT ON TABLE ref.tr_dataaccess_dta
IS 'Table with two values, Public or Restricted access.';
```

```
tr_dataaccess_dta <- dbGetQuery(con_diaspara_admin,
                                "SELECT * FROM refwgeel.tr_dataaccess_dta")

dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_dataaccess_dta
  SELECT * FROM refwgeel.tr_dataaccess_dta
  ;")#2
```

Table 2.16: Data access

dta_code	dta_description
Public	Public access according to ICES Data Policy
Restricted	Restricted access (wgeel find a definition)

2.14 Missing data (tr_missvalueqal_mis)

SQL code to create tables

```
--DROP TABLE IF EXISTS ref.tr_missvalueqal_mis CASCADE;

CREATE TABLE ref.tr_missvalueqal_mis(
   mis_code TEXT PRIMARY KEY,
   mis_description TEXT NOT NULL,
   mis_definition TEXT);

GRANT ALL ON ref.tr_missvalueqal_mis TO diaspara_admin;
GRANT SELECT ON ref.tr_missvalueqal_mis TO diaspara_read;

COMMENT ON TABLE ref.tr_missvalueqal_mis IS 'Table showing the qualification when value is m
```

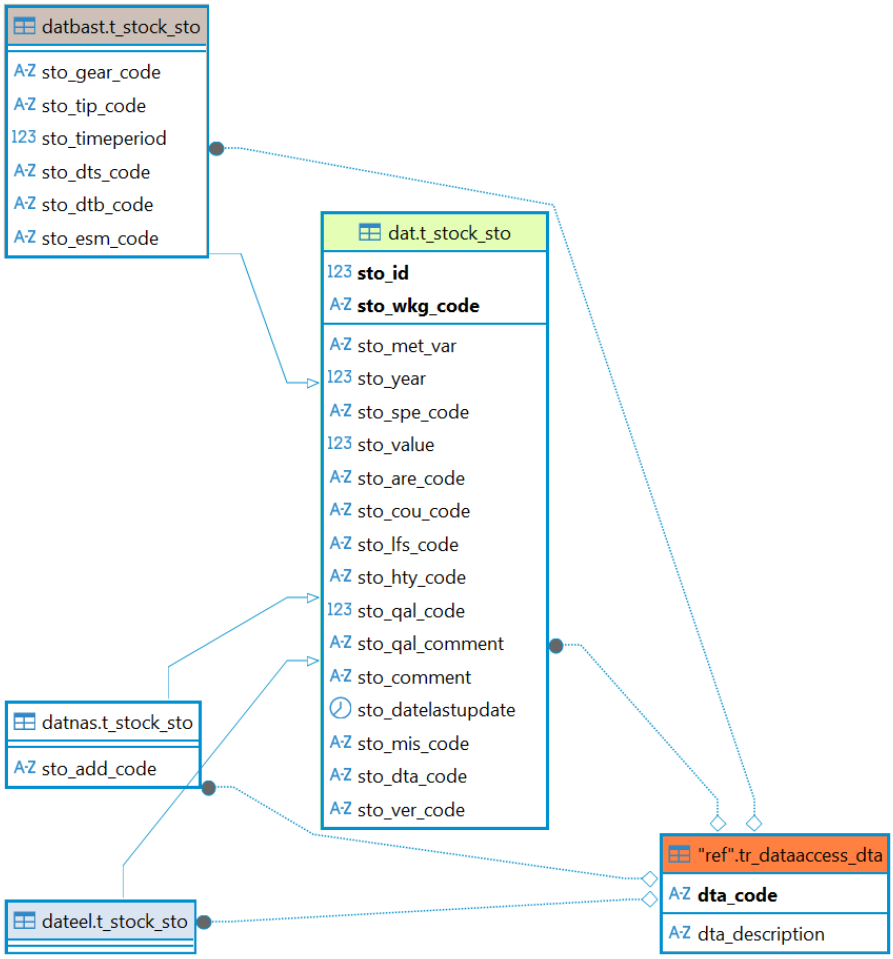


Figure 2.11: diagram of tr_dataaccess_dta

```
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_missvalueqal_mis
SELECT
'NR',
'Not reported',
'Data or activity exist but numbers are not reported to authorities (for example for commerc
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_missvalueqal_mis
SELECT
'NC',
'Not collected',
'Activity / habitat exists but data are not collected by authorities (for example where a fi
dbExecute(con_diaspara_admin,
          "INSERT INTO ref.tr_missvalueqal_mis
SELECT
'NP',
'Not pertinent',
'Where the question asked does not apply to the individual case (for example where catch dat
```

Table 2.17: Code fo

mis_code	mis_description	mis_definition
NR	Not reported	Data or activity exist but numbers are not reported to authorities (for examp
NC	Not collected	Activity / habitat exists but data are not collected by authorities (for examp
NP	Not pertinent	Where the question asked does not apply to the individual case (for example

TODO ICES
This is used in the t_stock_sto table by both WGEEL and WGBAST. Either a value is provided or this field has to be provided (conditional mandatory in the format).

2.15 Life stages (tr_lifestage_lfs)

Life stages cannot easily be shared among all species, they are species specific, probably similar between Sea trout and Salmon, but there is a large gap between a leptocephalus and a parr.

CREATE A STEERING GROUP IN WGDIAD
The following definitions have been checked by the working groups and a draft of this referential has been created in ICES : DIASPARA : Diadromous fish life stage #885. However, as for some other new referential created in ICES, they will need to be validated. It was proposed during the final DIASPARA meeting that this steering group is created within WGDIAD to discuss these issues.

2.15.1 Considerations about the database structure

For life stage, unlike in other referentials, using working-group-specific life stages would lead to confusion. WGBAST and WGNAS would share the same stages for salmon. So unlike in many other table, the referentials will not use inheritance (see paragraph Section 1 for more details on inheritance). This means that we will create a table grouping all life stages and then we will only select the relevant ones at working group levels. For instance currently WGEEL does not use the `Egg` or `Parr` stages. It will be listed in the `ref.tr_lifestage_lfs` table but not in the `refeel.tr_lifestage_lfs` table. So the working group referentials, `refnas`, `refbast`, `refeel` … will have `tr_lifestage_lfs` tables with a foreign key to `ref.tr_lifestage_lfs`, and a subset of values used by the working group.

2.15.2 The lifestages in working group databases

The creation of life stage is discussed in the issue 16 in git github link to issue

Stages use are described in WGNAS metadata paragraph life stage of the WGNAS description report and WGBAST reports (though for the "young fish" -data this concept is mixed with age) paragraph life stage of the WGBAST description report. So they are not used as a separate column to describe the data. But in the eel database they are and so we will need to add this dimension to the table structure.

Adding a stage dimension
Currently the stage is used by WGEEL, not by WGNAS and not directly in WGBAST. Indeed for WGBAST there is no column stage in landings data. There is an age column in the juvenile data used to describe the number released from hatchery at various stages (age in the juvenile database). Spatial information, and information about the age are used to split the dataset, not the stage, but the different datasets (landings, juvenile) do correspond to different stages. We have chosen to add stages everywhere and populate the column using the metadata even if for WGNAS there is no practical use the information is not used currently.

2.15.3 Use of lifestages to include other information

The different steps in the models are identified using spatio-temporal units and stages (e.g; post smolt and location at sea) ICES (2024a), a location and stage to descibe the different steps in the return migration (for instance returning adult in seawater, in freshwater…). Some of the elements added in the stage column are not stage per-se, but elements used to describe the spatio-temporal elements within the life-cycle, for instance the use PFA (pre-fishery-abundance) which correspond to the number at sea of different stages before fishery. Note that this parameter is no longer used, that is, it's in the metadata but the variable in metadata are no longer in the database (this stands for logN4, N4, logit_theta4, tau_theta4, theta4). To deal with these spatio-temporal elements that are not stage, we will simplify the stages table, remove elements which are not stages, and still refer to spatio temporal parameters from their definition. Here we are focusing on the stock DB which will group information, but the individual metric database might require more details that the simple adult stage. For this reason, we will add the maturity scale from ICES in the DB.

2.15.4 Adding a bit more complexity with the eel

In some cases, a mixture of stages are used. Many fyke net fisheries for eel will not distinguish between yellow and silver eel stage and WGEEL uses the `YS` stage. Some historical trap series did not distinguish between small yellow eels and glass eel, or the glass eel ascending are in a late stage. In that case the `GY`stage is used to count the recruits.

QUESTION TO WGEEL

The new database will have to include a source WILD/HATCHERY/AQUACULTURE shouldn't we use that opportunity to get rid of `OG` (ongrown) and `QG` (quarantine glass eel) which are cumbersome when we try to make simple graphs ?

Answer WGEEL (Cédric)

Stages OG and QG will be marked as deprecated for the stock, but they will probably be used in the release database that needs to be created for EDA.

2.15.5 Code to create the stage table

The code for creating `tr_lifestage_lfs` is shown below, it also includes the import of the WGEEL stage table.

SQL code to create table tr_lifestage_lfs

```
-- Table for lifestage
-- there is one table for all working groups
-- so this table is not inherited (otherwise two wkg could create referential for the same s
```

```
DROP TABLE IF EXISTS ref.tr_lifestage_lfs CASCADE;
CREATE TABLE ref.tr_lifestage_lfs (
  lfs_id SERIAL PRIMARY KEY,
  lfs_code TEXT NOT NULL UNIQUE,
  lfs_name TEXT NOT NULL,
  lfs_spe_code character varying(3) NOT  NULL,
  lfs_description TEXT,
  lfs_icesvalue character varying(4),
  lfs_icesguid uuid,
  lfs_icestablesource text,
CONSTRAINT fk_lfs_spe_code FOREIGN KEY (lfs_spe_code)
  REFERENCES ref.tr_species_spe(spe_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
CONSTRAINT uk_lfs UNIQUE (lfs_code, lfs_spe_code)
);

COMMENT ON TABLE ref.tr_lifestage_lfs IS 'Table of lifestages';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_id IS 'Integer, primary key of the table';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_code IS 'The code of lifestage';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_name IS 'The english name of lifestage';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_spe_code IS 'The code of the species referenced f
tr_species_spe : use aphiaID eg ''126281'' for eel ';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_description IS 'Definition of the lifestage';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_icesvalue IS 'Code for the lifestage in the ICES
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_icesguid IS 'GUID in the ICES database';
COMMENT ON COLUMN ref.tr_lifestage_lfs.lfs_icestablesource IS 'Source table in ICES vocab';


GRANT ALL ON ref.tr_lifestage_lfs TO diaspara_admin;
GRANT SELECT ON ref.tr_lifestage_lfs TO diaspara_read;
```

2.15.6 Existing stages in ICES dictionaries

You can run the following code to see the candidate tables in ICES, in summary: none fitted and this part is skipped to shorten the report.

QUESTION TO ICES

Does it make sense to use a code outside from its scope (I seems to me that yes). If so can we use the TS_DevStage and add values in it and propose definitions ?

Table 2.18: ICES vocabularies for life stages

```
types <- icesVocab::getCodeTypeList()
types[grep('stage', tolower(types$Description)),]|> kable()
TS_MATURITY <- icesVocab::getCodeList('TS_MATURITY')
TS_DevStage <- icesVocab::getCodeList('TS_DevStage')
# Devstage is 1 to 15 => no use
DevScale <- icesVocab::getCodeList('DevScale')
# At the present the codetypes target Eggs and Larvae surveys
# This is a description of scales types using different publications => not for use()
kable(TS_MATURITY, caption = "TS_MATURITY") |> kable_styling(bootstrap_options = c("striped"
kable(TS_DevStage, caption = "Devstage scale, this seems to be about shrimp (berried seems t
```

2.15.7 Importing the lifestages for Salmon

Some of the definitions come from Ontology portal.

```
# below the definition of Smolt, post-smolt and Adult provided by Etienne.
lfs <- tribble(
  ~lfs_code, ~lfs_name, ~lfs_spe_code, ~lfs_description,
  ~lfs_icesvalue, ~lfs_icesguid, ~lfs_icestablesource,
  "E",   "Egg", "127186" , "In fish, the term egg usually refers to female haploid gametes."
  "E",   "0424ae90-03aa-4e73-8cda-e8745d0b8158", "TS_DevStage",
  "EE",   "Eyed egg", "127186" , "Eyed eggs are fertilized eggs that have developed to the s
  NA,  NA , NA,
  "ALV",   "Alevin with the yolk sac", "127186" , "Larval salmon that have hatched but have
  NA,  NA , NA,
  "FR",   "Fry", "127186" , "A young salmonid at the post-larval stage who has resorbed the
  NA,  NA , NA,
  "P", "Parr", "127186",
  "A young salmonid with parr-marks before migration to the sea and after dispersal from the
  NA, NA, NA,
  "SM", "Smolt", "127186", "A young salmonid which has undergone the transformation to adapt
  NA, NA, NA,
  "PS", "Post Smolt", "127186", "A salmonid at sea, after its migration to the sea as smolt.
  NA, NA, NA,
  "A", "Adult", "127186", " Salmonids that have fully developed morphological and meristic c
  "AL", "All stages", "127186", "All life stages are concerned.", NA, NA, NA,
  "_", "No life stage", "127186", "Reserved when the life stage makes no sense for the varia
)
dbWriteTable(con_diaspara_admin, "temp_lfs", lfs, overwrite = TRUE)
dbExecute(con_diaspara_admin, "DELETE FROM ref.tr_lifestage_lfs;")#24
dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_lifestage_lfs
    ( lfs_code, lfs_name, lfs_spe_code, lfs_description, lfs_icesvalue, lfs_icesguid, lfs_ic
    SELECT
     lfs_code,
```

```
     lfs_name,
     lfs_spe_code,
     lfs_description,
     lfs_icesvalue,
     lfs_icesguid::uuid,
     lfs_icestablesource
     FROM temp_lfs;")
dbExecute(con_diaspara_admin, "DROP TABLE temp_lfs")
```

2.15.8 Import lifestages for eel

```
dbExecute(con_diaspara_admin,"DELETE FROM ref.tr_lifestage_lfs WHERE lfs_spe_code ='126281';
dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_lifestage_lfs (lfs_code,lfs_name,lfs_descr
SELECT lfs_code,initcap(lfs_name),lfs_definition, '126281'
FROM refwgeel.tr_lifestage_lfs ;") # 8
```

2.15.9 Import lifestages for trutta

```
dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_lifestage_lfs
    (lfs_code, lfs_name, lfs_spe_code, lfs_description, lfs_icesvalue, lfs_icesguid, lfs_ice
    SELECT
    lfs_code,
    lfs_name,
    '127187' AS lfs_spe_code,
    lfs_description,
    lfs_icesvalue,
    lfs_icesguid::uuid,
    lfs_icestablesource
    FROM ref.tr_lifestage_lfs WHERE lfs_spe_code = '127186';")
```

WGTRUTTA Comment (Iain Malcolm)

In terms of stage, we would need alevin, fry (YoY, 0+), parr (>0+). In the Marine Directorate database lifestage is called “field recorded lifestage” to separate from lifestage derived from ageing by scale reading.

> DIASPARA : OK this seems to fit to our current vocabulary. For scale reading we have prepared a different field in the individual metric database. It will not be used in the stock database.

2.15.10 Content of the tr_lifestage_lfs table

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_lifestage_lfs where lfs_spe_code = '127186';"
knitr::kable()|>
 kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

lfs_id	lfs_code	lfs_name	lfs_spe_code	lfs_description
50	E	Egg	127186	In fish, the term egg usually refers to female
51	EE	Eyed egg	127186	Eyed eggs are fertilized eggs that have develo
52	ALV	Alevin with the yolk sac	127186	Larval salmon that have hatched but have no
53	FR	Fry	127186	A young salmonid at the post-larval stage wh
54	P	Parr	127186	A young salmonid with parr-marks before mi
55	SM	Smolt	127186	A young salmonid which has undergone the t
56	PS	Post Smolt	127186	A salmonid at sea, after its migration to the
57	A	Adult	127186	Salmonids that have fully developed morphol
58	AL	All stages	127186	All life stages are concerned.
59	_	No life stage	127186	Reserved when the life stage makes no sense

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_lifestage_lfs where lfs_spe_code = '126281';"
knitr::kable()|>
 kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

lfs_id	lfs_code	lfs_name	lfs_spe_code	lfs_description
60	G	Glass Eel	126281	Young, unpigmented eel, recruiting from the se
61	S	Silver Eel	126281	Migratory phase following the yellow eel phase
62	GY	Glass Eel + Yellow Eel	126281	A mixture of glass and yellow eel, some traps
63	OG	Ongrown Eel	126281	Eel that have been held in water tanks for som
64	QG	Quarantined Eel	126281	Ongrown eel (see definition above) that have b
65	AL	All Stages	126281	All stages combined
66	YS	Yellow Eel+ Silver Eel	126281	Yellow and Silver eel defined below
67	Y	Yellow Eel	126281	Life-stage resident in continental waters. Ofte

Follow up in ICES vocab

You can follow up this issue in ICES DIASPARA: Diadromous fish life stages #885

2.16 Maturity (ts_maturity_mat)

Working on stages, and looking at the ICES vocab, we have decided to include information on maturity.

SQL code to create table `tr_maturity_mat`

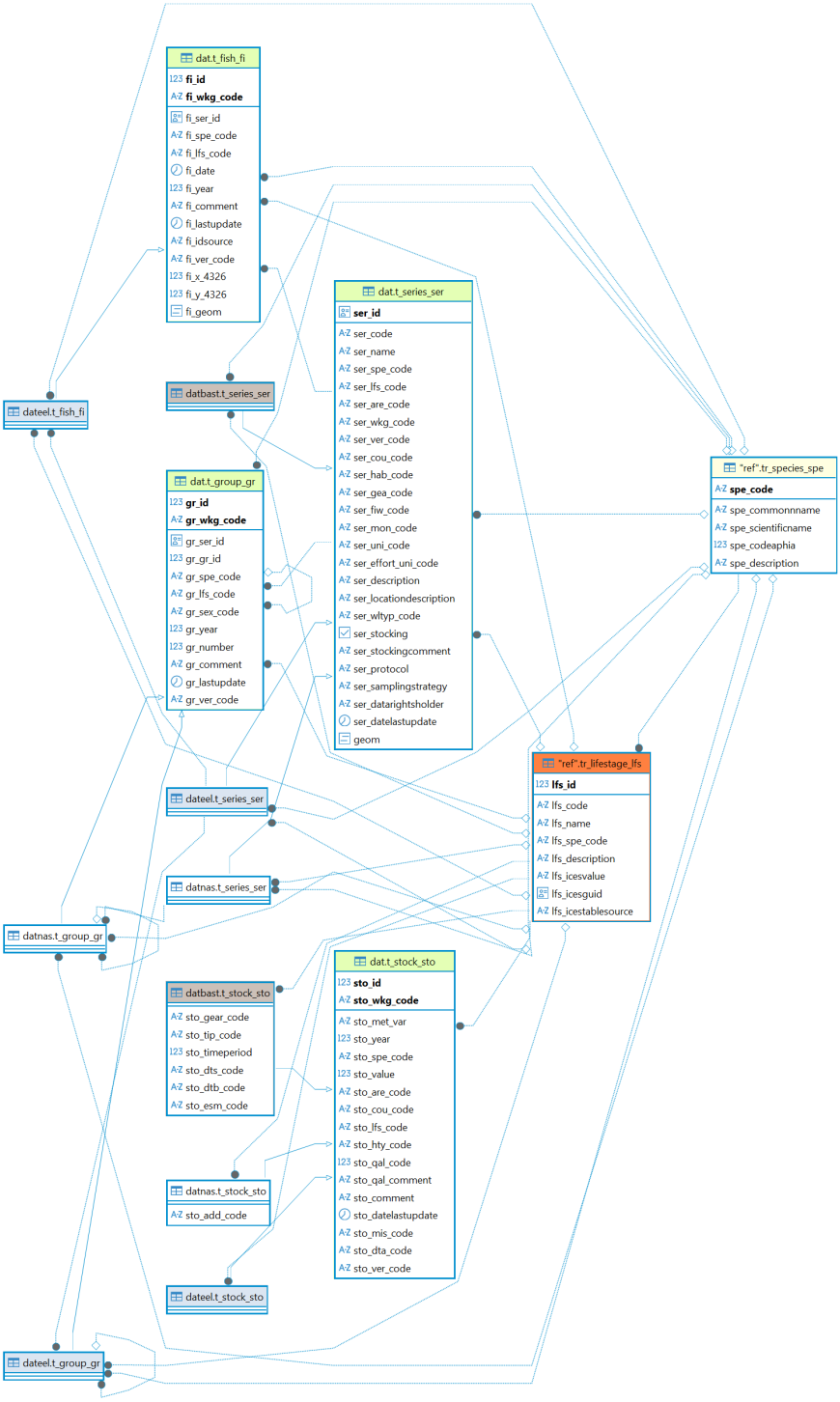


Figure 2.12: diagram of tr_lifestage_lfs

```
-- maturity table code


DROP TABLE IF EXISTS ref.tr_maturity_mat CASCADE;
CREATE TABLE ref.tr_maturity_mat (
  mat_id SERIAL PRIMARY KEY,
  mat_code TEXT NOT NULL CONSTRAINT uk_mat_code UNIQUE,
  mat_description TEXT,
  mat_icesvalue character varying(4),
  mat_icesguid uuid,
  mat_icestablesource text
);

COMMENT ON TABLE ref.tr_maturity_mat IS 'Table of maturity corresponding to the 6 stage scal
COMMENT ON COLUMN ref.tr_maturity_mat.mat_id IS 'Integer, primary key of the table';
COMMENT ON COLUMN ref.tr_maturity_mat.mat_code IS 'The code of maturity stage';
COMMENT ON COLUMN ref.tr_maturity_mat.mat_description IS 'Definition of the maturity stage';
COMMENT ON COLUMN ref.tr_maturity_mat.mat_icesvalue IS 'Code (Key) of the maturity in ICES c
COMMENT ON COLUMN ref.tr_maturity_mat.mat_icesguid IS 'UUID (guid) of ICES, you can access t
COMMENT ON COLUMN ref.tr_maturity_mat.mat_icestablesource IS 'Source table in ICES';
GRANT ALL ON ref.tr_maturity_mat TO diaspara_admin;
GRANT SELECT ON ref.tr_maturity_mat TO diaspara_read;
```

In Salmoglob, information about the stage mixes information on stage and maturity. The use of SMSF vocabulary has been made mandatory for all countries since 2020 and WGBIOP (ICES 2024b) has revised the referential and emphasized the need of its use for a consistent stock assessment.

In the report the stages are defined as following in Maturitstage.

State	Stage	Possible sub-stages
SI. Sexually immature	A. Immature	
SM. Sexually mature	B. Developing	Ba. Developing but functionally immature (first-time developer) Bb. Developing and functionally mature
	C. Spawning	Ca. Actively spawning Cb. Spawning
	D. Regressing/Regenerating	Da. Regressing Db. Regenerating
	E. Omitted spawning	
	F. Abnormal	

Table SMSF (WKMATCH 2012 maturity scale revised). Source: [@ices_wb-biop_2024].

Of note the following comments by WGBIOP :

- The substage Ba identifies a sexually mature but functionally immature (virgin developing for the first time) fish which is not going to contribute to the current upcoming spawning season. Either it is uncertain if the fish will make it for the upcoming spawning season as it is a long time to the current upcoming spawning season (i.e. if maturity is assessed 8 months prior to the spawning season it is unsure if the first time developer will be ready to spawn in 8 months time), or the time between assessing the maturity stage and the current upcoming spawning season is too short to fully develop the oocytes (i.e. if it takes 6 months to fully develop oocytes from previtellogenic to eggs and a Ba fish is found 3 months prior to the current upcoming spawning season, it will not have enough time to develop the oocytes).
- the substage Bb identifies a developing and functionally mature (first or repeat spawner!!) fish which, in most of the cases is going to contribute to the current spawning season. This stage has visible oocytes and grainy appearance of the gonads on the macroscopic scale, and vitellogenic oocytes on the histological key.

Following this report and the comments made by ICES data center the following codes are proposed (Table Table 2.22).

```
maturity <- icesVocab::getCodeList('TS_MATURITY')

maturity <- maturity |>
  rename("mat_icesguid"="GUID",  "mat_icesvalue" = "Key", "mat_description" = "Description")
  select ( mat_icesvalue, mat_icesguid, mat_description) |>
  filter(mat_icesvalue %in% c("A","B","Ba","Bb","C","Ca","Cb","D","Da","Db","E", "F")) |>
  mutate(mat_icestablesource = "TS_MATURITY",
         mat_id = 1:12,
         mat_code = mat_icesvalue) |>
  select(mat_id, mat_code, mat_description, mat_icesvalue, mat_icesguid, mat_icestablesource

DBI::dbWriteTable(con_diaspara_admin, "temp_maturity", maturity, overwrite = TRUE)


DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_maturity_mat
(mat_id, mat_code, mat_description, mat_icesvalue, mat_icesguid, mat_icestablesource)
SELECT mat_id, mat_code, mat_description, mat_icesvalue, mat_icesguid::uuid, mat_icestablesc
FROM temp_maturity")# 12


DBI::dbExecute(con_diaspara_admin, "DROP table temp_maturity")
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_maturity_mat;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.22: Table of maturity codes

mat_id	mat_code	mat_description	mat_icesvalue	mat_
1	A	Immature	A	a25f8
2	B	Developing	B	62b0
3	Ba	Developing, but functionally immature (first-time developer)	Ba	4b46
4	Bb	Developing and functionally mature	Bb	5fb2
5	C	Spawning	C	c5e1
6	Ca	Actively spawning	Ca	a9bc
7	Cb	Spawning capable	Cb	8d87
8	D	Regressing / Regenerating	D	5af7
9	Da	Regressing	Da	c111
10	Db	Regenerating	Db	ecb4
11	E	Omitted spawning	E	a1cf8
12	F	Abnormal	F	1736

Add maturity to metadata in WGNAS

We still have to modify the metadata in WGNAS, create a new column for maturity and link it. This has to be done by WGNAS : Metadata : integrate the maturity referential #27

2.17 Habitat type (tr_habitattype_hty)

This table (Table 2.23) is used in RDBES, and those stages are consistent with WGBAST and WGEEL. When creating habitat types for WGEEL, we tried to follow the ICES vocab at the time, so it's mostly similar except that `C` (coastal) in WGEEL is `C (WFD Coastal water)` in WLTYP but is also reported as `MC (Marine Coastal)` in the RDBES and freshwater is `F` instead of `FW`. WGBAST separates Marine Open `O` (instead of `MO`), Marine coastal `C` (instead of `MC`) and rivers `R` (instead of `FW`). In the report it is said that `S` sea is used when it is not possible to distinguish between coastal and marine open, but the code is not in the database (If I'm not wrong see WGBAST database description - catch habitat). Other elements in this vocab will not be used (e.g. `TT`, `Beach`, …). Currently the RDBES uses the following codes from `FW Fresh water`, `MC Marine water (coast)`, `MO Marine water (open sea)`, `MC Marine water (coast)` and `NA Not applicable`.

QUESTION TO ICES (Maria, Joana, Henrik)

Why has the code `MC` been chosen instead of `C` for RDBES ? What is the rationale for not using the WFD ? Is it for non European countries Eel mostly follows the WFD (in EU countries) as the units should be based on river basins. What should we use there ? Our choice there would be to use `MC`, `MO`, `T` and `FW` and so add `T` to the list of vocabularies used by RDBES, would you agree ?

```
WLTYP <- icesVocab::getCodeList('WLTYP')

# At the present the codetypes target Eggs and Larvae surveys
# This is a description of scales types using different publications => not for use()
kable(WLTYP, caption = "WLTYP") |> kable_styling(bootstrap_options = c("striped", "hover", '
```

Id	Guid	Key	Description
156363	9388a49e-4a5f-46b3-9b7b-8dcfb03e7b6f	BP	Beach - peri-urban
156362	a6e1119b-3bcc-4b0a-8053-f8f7a86f0053	BR	Beach - rural
156361	53708e1d-d3c6-47a7-a0ee-1c70e6fa7f9b	BU	Beach - urban
53327	86d9897d-6adc-430e-a689-f321b75cadcc	C	WFD Coastal water
53329	0a28de1c-018a-4841-b3a3-a68e01c81cd5	CE	Coastal water (Estuary)
54066	85a228c9-9033-4f97-accf-4968db78a90a	CF	Coastal water (Fjord)
53330	e57989c4-a3f9-4560-9460-60dce3a26185	CR	Coastal water (River)
252557	eb979616-8b95-4c1b-92e8-b5f8d8bf1b96	FW	Fresh water
53334	c557dc19-27b9-46b6-a164-d7d8d4f55738	L	Land station
134991	5b3da387-3b2b-47c4-9967-c3161f533207	LK	Lake
252556	b51355b0-b905-4b4e-ab12-98d9b47d7752	MC	Marine water (coast)
53333	a75522ef-5e4a-4e2d-8550-38091cb6c994	MO	Marine water (open sea)
252558	913ae617-a160-4687-a62c-8923c6762c4f	NA	Not applicable
53331	1c792737-6f55-422a-b709-88af2d78c4ea	T	WFD Transitional water - implies reduced salini
53332	4f85f72f-c2f0-41c7-b966-c80c897b80d7	TT	Transitional water (Tidal) - significant tide and

2.17.1 Code to create the habitat type table.

The code for creating `tr_habitat_type` is shown below.

SQL code to create table `tr_habitat_type`

```
-- Table for habitattype
-- there is one table for all working groups
-- This mostly only imports some of the codes from the WLTYP vocab


DROP TABLE IF EXISTS ref.tr_habitattype_hty CASCADE;
```

```
CREATE TABLE ref.tr_habitattype_hty (
  hty_id SERIAL PRIMARY KEY,
  hty_code TEXT NOT NULL UNIQUE,
  hty_description TEXT,
  hty_icesvalue character varying(4),
  hty_icesguid uuid,
  hty_icestablesource text
);

COMMENT ON TABLE ref.tr_habitattype_hty IS 'Table of habitat types, takes from the WLTYP voc
COMMENT ON COLUMN ref.tr_habitattype_hty.hty_id IS 'Integer, primary key of the table';
COMMENT ON COLUMN ref.tr_habitattype_hty.hty_code IS 'The code of the habitat';
COMMENT ON COLUMN ref.tr_habitattype_hty.hty_description IS 'Definition of the lifestage';
COMMENT ON COLUMN ref.tr_habitattype_hty.hty_icesvalue IS 'Code for the lifestage in the ICE
COMMENT ON COLUMN ref.tr_habitattype_hty.hty_icesguid IS 'GUID in the ICES database';
COMMENT ON COLUMN ref.tr_habitattype_hty.hty_icestablesource IS 'Source table in ICES vocab'


GRANT ALL ON ref.tr_habitattype_hty TO diaspara_admin;
GRANT SELECT ON ref.tr_habitattype_hty TO diaspara_read;
```

2.17.2 Import the habitat type

The codes are imported in Table Table 2.24.

```
habitat <- icesVocab::getCodeList('WLTYP')

habitat <- habitat |>
  rename("hty_icesguid"="GUID",  "hty_icesvalue" = "Key", "hty_description" = "Description")
  select ( hty_icesvalue, hty_icesguid, hty_description) |>
  filter(hty_icesvalue %in% c("MC","MO","FW","T")) |>
  mutate(hty_icestablesource = "TS_habitat",
         hty_id = 1:4,
         hty_code = hty_icesvalue) |>
  select(hty_id, hty_code, hty_description, hty_icesvalue, hty_icesguid, hty_icestablesource

DBI::dbWriteTable(con_diaspara_admin, "temp_habitat", habitat, overwrite = TRUE)


DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_habitattype_hty
(hty_id, hty_code, hty_description, hty_icesvalue, hty_icesguid, hty_icestablesource)
SELECT hty_id, hty_code, hty_description, hty_icesvalue, hty_icesguid::uuid, hty_icestablesc
FROM temp_habitat")# 4
```

```
DBI::dbExecute(con_diaspara_admin, "DROP table temp_habitat")
```

- The following table is proposed (Table Table 2.24).

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_habitattype_hty;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.24: Table of habitat types

hty_id	hty_code	hty_description	hty_icesvalue	hty_icesguid
1	FW	Fresh water	FW	eb979616-8b95-4
2	MC	Marine water (coast)	MC	b51355b0-b905-4
3	MO	Marine water (open sea)	MO	a75522ef-5e4a-4e
4	T	WFD Transitional water - implies reduced salinity	T	1c792737-6f55-42

2.18 Quality (tr_quality_qal)

This code is used by wgeel. Currently the WGNAS uses an archive table, WGEEL uses historical data with a different quality ID. Both have chosen never to remove any data. It does not look like these procedures, handled by shiny app, are compatible with ICES procedures.

QUESTION TO ICES

WGNAS uses an archive table for historical data. WGEEL uses a code to "deprecate" old values.

- In practise for WGNAS it means that each time a new row replaces an old one, the data is saved in an archive table, with the same structure as the main data table, but with the name of the people who have handled the change and the date. The version is replaced with a new number in the `database` table.
- For WGEEL, all the rows are kept in the same table. Historical value get a code like 18, ..., 25 which identifies the year that the line was removed. All data submitted to the database are kept, so if during a datacall, a new value is submitted that is a duplicate from an old one, then the user in the shiny has to edit an excel table saying which value he wants to keep. If for instance he wants to keep the old value, then the new row will go into the database with a qal_id 25 if the change is made in 2025. The table Table 2.25 is used.

How do you work in ICES to keep historical data ?

Answer ICES (Carlos)

Normally we don't ....

We need an archive

It was further agreed during the final meeting and several other informal meetings that an archive format will be created to adapt to the needs of the Working groups (shiny using the archive table to check for changes …). Currently historical values are kept in WGEEL, and we have kept the existing codes 18, 19 … These data will be removed at the end and we will see how we deal with this when fully transfering to ICES database. Probably we will keep some table with in our records (WGEEL).

Table tr_quality_qal used by the wgeel

Table 2.25

qal_id	qal_level	qal_text
1	good quality	the data passed the quality checks of the wgeel
2	modified	The wgeel has modified that data
4	warnings	The data is used by the wgeel, but there are warnings on its quality (see comments)
0	missing	missing data
3	bad quality	The data has been judged of too poor quality to be used by the wgeel, it is not used
18	discarded_wgeel_2018	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2018
19	discarded_wgeel_2019	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2019

Table 2.25

qal_id	qal_level	qal_text
20	discarded_wgeel_2020	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2020
21	discarded_wgeel_2021	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2021
-21	discarded 2021 biom mort	This data has either been removed from the database in favour of new data, this has been done systematically in 2021 for biomass and mortality types
22	discarded_wgeel_2022	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2022
23	discarded_wgeel 2023	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2023
24	discarded_wgeel 2024	This data has either been removed from the database in favour of new data, or corresponds to new data not kept in the database during datacall 2024

ICES (Maria)

We have this quality flag code type from seadatanet.

ANSWER DIASPARA

OK we need to adapt there - 0 in wgeel becomes 9 - 1 is OK - 2 becomes 5 - 3 becomes 4 - 4 becomes 3

Values other than 0 1 2 3 4 will have to be ignored? or considered as historical ? Currently I'm keeping them but we'll probably copy all those lines in another table kept only by WGEEL. Figure Figure 2.13 shows which tables will be updated after the change.

2.18.1 Code to create the table

The code for creating `tr_quality_qal` is shown below.

SQL code to create table `tr_quality_qal`

```
-- Table for quality

DROP TABLE IF EXISTS ref.tr_quality_qal CASCADE;
CREATE  TABLE ref.tr_quality_qal (
  qal_code int4 NOT NULL,
  qal_description text NULL,
  qal_definition text NULL,
  qal_kept bool  NULL,
    CONSTRAINT tr_quality_qal_pkey PRIMARY KEY (qal_code)
);
COMMENT ON TABLE ref.tr_quality_qal IS 'Table of quality rating, 1 = good quality, 2 = modif
COMMENT ON COLUMN ref.tr_quality_qal.qal_code IS 'Data quality code';
COMMENT ON COLUMN ref.tr_quality_qal.qal_description IS 'Data quality description';
COMMENT ON COLUMN ref.tr_quality_qal.qal_definition IS 'Definition of the quality code';
COMMENT ON COLUMN ref.tr_quality_qal.qal_kept IS 'Are the data with this score kept for anal

GRANT ALL ON ref.tr_quality_qal TO diaspara_admin;
GRANT SELECT ON ref.tr_quality_qal TO diaspara_read;
```

2.18.2 Import the quality code

The codes are imported in Table 2.26 using the following code :

```
 SDN_FLAGS<- icesVocab::getCodeList('SDN_FLAGS')
dbWriteTable(conn = con_diaspara,name = "temp_sdn_flags", value = SDN_FLAGS)
```

```
# old code deprecated, see next sql
# dbExecute(con_diaspara_admin,"DELETE FROM ref.tr_quality_qal;")
# dbExecute(con_diaspara, "ALTER TABLE ref.tr")
#
# dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_quality_qal (qal_code, qal_description,
# SELECT qal_id, qal_level, qal_text, qal_kept FROM refwgeel.tr_quality_qal ;")
# # 13
# # changing one definition linked to wgeel
# dbExecute(con_diaspara_admin, "UPDATE ref.tr_quality_qal set qal_definition = 'the data pa
# # This one only causes problems.... Remove.
# dbExecute(con_diaspara_admin, "DELETE FROM ref.tr_quality_qal WHERE qal_code = 0")
```

SQL code to modify table `tr_quality_qal`

```
-- See report following Maria's comment there is a referential, here is the script to adapt

ALTER TABLE ref.tr_quality_qal  ADD COLUMN qal_icesvalue CHARACTER VARYING (4);
ALTER TABLE ref.tr_quality_qal  ADD COLUMN qal_icesguid uuid ;
ALTER TABLE ref.tr_quality_qal  ADD COLUMN qal_icestablesource TEXT ;

-- table public.temp_sdn_flags has been inserted in R
-- see report https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p7stock/stockdb.


/*
 * 0 has been removed from the table
 */

SELECT * FROM public.temp_sdn_flags WHERE "Key" = '9';
DELETE FROM  ref.tr_quality_qal WHERE qal_code = 9;
INSERT INTO ref.tr_quality_qal (qal_code, qal_description, qal_definition,
qal_kept, qal_icesvalue, qal_icesguid, qal_icestablesource)
 SELECT "Key"::integer AS qal_code,
  'Missing' AS qal_description,
 "Description" AS qal_definition,
 FALSE AS qal_kept,
 "Key" AS qal_icesvalue,
 "Guid"::UUID AS qal_icesguid,
 'SDN_FLAGS' AS qal_icestablesource
 FROM public.temp_sdn_flags WHERE "Key" = '9'; --1


 /*
  * 1 is OK
  */
```

```
SELECT * FROM public.temp_sdn_flags WHERE "Key" = '1';
UPDATE ref.tr_quality_qal SET (qal_description, qal_definition, qal_kept,
qal_icesvalue, qal_icesguid, qal_icestablesource) =
(s.qal_description, s.qal_definition, s.qal_kept, s.qal_icesvalue,
s.qal_icesguid, s.qal_icestablesource)
FROM (
 SELECT
  'Good quality' AS qal_description,
 "Description" AS qal_definition,
 TRUE AS qal_kept,
 "Key" AS qal_icesvalue,
 "Guid"::UUID AS qal_icesguid,
 'SDN_FLAGS' AS qal_icestablesource
 FROM public.temp_sdn_flags WHERE "Key" = '1') AS s
 WHERE qal_code = 1;

  /*
  * 2 becomes 5
  */
DELETE FROM  ref.tr_quality_qal WHERE qal_code = 5;
UPDATE ref.tr_quality_qal SET (qal_code, qal_description, qal_definition,
qal_kept, qal_icesvalue, qal_icesguid, qal_icestablesource) =
(s.qal_code, s.qal_description, s.qal_definition, s.qal_kept,
s.qal_icesvalue, s.qal_icesguid, s.qal_icestablesource)
 FROM (
 SELECT
  "Key"::integer AS qal_code,
  'Modified' AS qal_description,
 "Description"|| '. Warning this was 2 previously in the WGEEL database.' AS qal_definition,
 TRUE AS qal_kept,
 "Key" AS qal_icesvalue,
 "Guid"::UUID AS qal_icesguid,
 'SDN_FLAGS' AS qal_icestablesource
 FROM public.temp_sdn_flags WHERE "Key" = '5') AS s
 WHERE tr_quality_qal.qal_code = 2; --1

   /*
  * 3 becomes 100
  * 4 becomes 3
  * 100 becomes 4
  */



 UPDATE ref.tr_quality_qal SET (qal_code, qal_description, qal_definition,
```

```
qal_kept, qal_icesvalue, qal_icesguid, qal_icestablesource) =
(s.qal_code, s.qal_description, s.qal_definition, s.qal_kept,
s.qal_icesvalue, s.qal_icesguid, s.qal_icestablesource)
FROM
(SELECT
100 AS qal_code,
 qal_description,
 "Description"||'. Previously 3 in WGEEL with definition:  '||qal_definition AS qal_definiti
 qal_kept,
 "Key" AS qal_icesvalue,
 "Guid"::UUID AS qal_icesguid,
 'SDN_FLAGS' AS qal_icestablesource
 FROM ref.tr_quality_qal, temp_sdn_flags
 WHERE qal_code = 3 AND  "Key" = '4')s
 WHERE tr_quality_qal.qal_code = 3;


 UPDATE ref.tr_quality_qal SET (qal_code, qal_description, qal_definition,
qal_kept, qal_icesvalue, qal_icesguid, qal_icestablesource) =
(s.qal_code, s.qal_description, s.qal_definition, s.qal_kept,
s.qal_icesvalue, s.qal_icesguid, s.qal_icestablesource)
FROM
(SELECT
 3 AS qal_code,
 qal_description,
 "Description"||'. Previously 4 in WGEEL with definition: '||qal_definition AS qal_definitio
 qal_kept, "Key" AS qal_icesvalue,
 "Guid"::UUID AS qal_icesguid,
 'SDN_FLAGS' AS qal_icestablesource
 FROM ref.tr_quality_qal, temp_sdn_flags
 WHERE qal_code = 4 AND  "Key" = '3')s
 WHERE tr_quality_qal.qal_code = 4;


 UPDATE ref.tr_quality_qal SET qal_code = 4 WHERE qal_code =100;




  /*
  * Insert values 2, 6, 7, 8 (I'm not using letters) as qal_code is an integer
  */


 INSERT INTO ref.tr_quality_qal (qal_code, qal_description, qal_definition,
```

```
qal_kept, qal_icesvalue, qal_icesguid, qal_icestablesource)
 SELECT "Key"::integer AS qal_code,
  'Missing' AS qal_description,
 "Description" AS qal_definition,
 TRUE AS qal_kept,
 "Key" AS qal_icesvalue,
 "Guid"::UUID AS qal_icesguid,
 'SDN_FLAGS' AS qal_icestablesource
 FROM public.temp_sdn_flags WHERE "Key" in ('2', '6', '7', '8'); --4

UPDATE "ref".tr_quality_qal
  SET qal_description='probably good'
  WHERE qal_code=2;
UPDATE "ref".tr_quality_qal
  SET qal_description='below detection'
  WHERE qal_code=6;
UPDATE "ref".tr_quality_qal
  SET qal_description='above detection'
  WHERE qal_code=7;
UPDATE "ref".tr_quality_qal
  SET qal_description='interpolated'
  WHERE qal_code=8;
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_quality_qal;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

qal_code	qal_description	qal_definition
18	discarded_wgeel_2018	This data has either been removed from the database in favour of n
19	discarded_wgeel_2019	This data has either been removed from the database in favour of n
20	discarded_wgeel_2020	This data has either been removed from the database in favour of n
21	discarded_wgeel_2021	This data has either been removed from the database in favour of n
-21	discarded 2021 biom mort	This data has either been removed from the database in favour of n
22	discarded_wgeel_2022	This data has either been removed from the database in favour of n
23	discarded_wgeel 2023	This data has either been removed from the database in favour of n
24	discarded_wgeel 2024	This data has either been removed from the database in favour of n
9	Missing	Missing value. The data value is missing. Any accompanying value
1	Good quality	Good value. Good quality data value that is not part of any identif
5	Modified	Changed value. Data value adjusted during quality control. Best pr
7	Above detection	Value in excess. The level of the measured phenomena was too larg
8	Interpolated	Interpolated value. This value has been derived by interpolation fro
3	Warnings	Probably bad value. Data value recognised as unusual during qualit

4	Bad quality	Bad value. An obviously erroneous data value.. Previously 3 in WG
2	Probably good	Probably good value. Data value that is probably consistent with r
6	Below detection	Value below detection. The level of the measured phenomena was t

⚠ To WGEEL values 3 (bad quality) and 4 (Warnings) have been inverted ! This means that all scripts will have to be adapted. Also 2 (modified by wgeel becomes 5)

2.19 Age (tr_age_age)

The code for creating `tr_age_age` (Table Table 2.27) is shown below.

SQL code to create table `tr_age_age`

```
-- Table for age
-- there is one table for all working groups
-- so this table is not inherited (otherwise two wkg could create referential for the same s

DROP TABLE IF EXISTS ref.tr_age_age;
CREATE TABLE ref.tr_age_age (
age_value INTEGER,
age_envir TEXT NOT NULL,
CONSTRAINT ck_age_envir CHECK (age_envir='Seawater' OR age_envir='Freshwater'),
age_code varchar(3) PRIMARY KEY,
age_description TEXT,
age_definition TEXT,
age_icesvalue character varying(4),
age_icesguid uuid,
age_icestablesource text);

ALTER TABLE ref.tr_age_age OWNER TO diaspara_admin;
GRANT SELECT ON ref.tr_age_age  TO diaspara_read;


INSERT INTO ref.tr_age_age VALUES (0, 'Freshwater', '0FW', '0 year in freshwater','Age of ju
INSERT INTO ref.tr_age_age VALUES (1, 'Freshwater', '1FW', '1 year in freshwater',NULL);
INSERT INTO ref.tr_age_age VALUES (2, 'Freshwater', '2FW', '2 years in freshwater',NULL);
INSERT INTO ref.tr_age_age VALUES (3, 'Freshwater', '3FW', '3 years in freshwater',NULL);
INSERT INTO ref.tr_age_age VALUES (4, 'Freshwater', '4FW', '4 years in freshwater',NULL);
INSERT INTO ref.tr_age_age VALUES (5, 'Freshwater', '5FW', '5 years in freshwater',NULL);
INSERT INTO ref.tr_age_age VALUES (6, 'Freshwater', '6FW', '6 years in freshwater',NULL);
INSERT INTO ref.tr_age_age VALUES (1, 'Seawater', '1SW', '1 year in seawater',NULL);
INSERT INTO ref.tr_age_age VALUES (2, 'Seawater', '2SW', '2 years in seawater',NULL);
INSERT INTO ref.tr_age_age VALUES (NULL, 'Seawater', 'MSW', 'Two years or more in seawater',
```

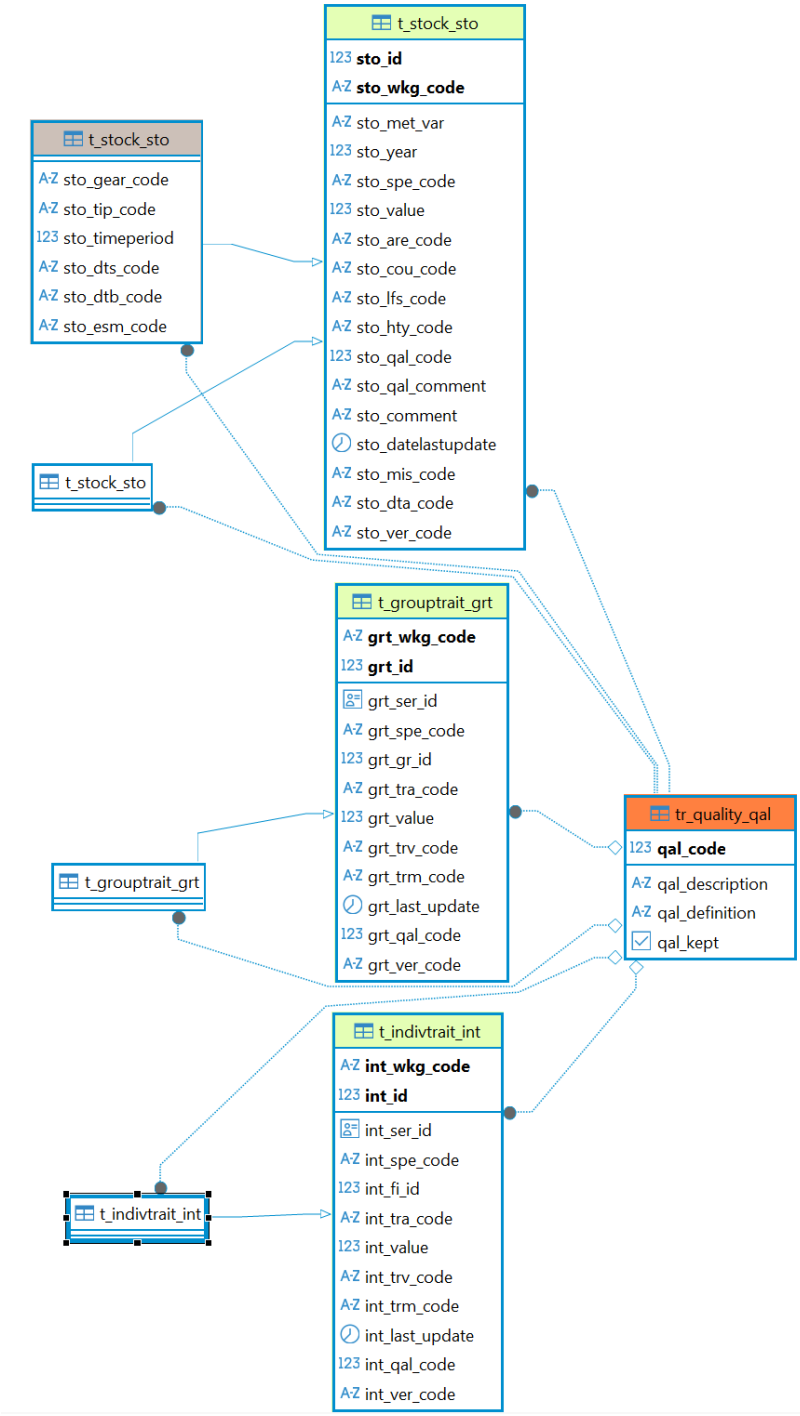


Figure 2.13: diagram of tr_quality_qal

```
INSERT INTO ref.tr_age_age VALUES (NULL, 'Seawater', 'MSW', 'Two years or more in seawater',
INSERT INTO "ref".tr_age_age (age_envir,age_code,age_description,age_definition)
  VALUES ('Freshwater','1+','Older than one year in freshwater','Groups all fishes older tha


COMMENT ON TABLE ref.tr_age_age IS 'Table of ages for salmonids';
COMMENT ON COLUMN ref.tr_age_age.age_value IS 'Integer, value of the age as integer';
COMMENT ON COLUMN ref.tr_age_age.age_envir IS 'Freshwater or Seawater';
COMMENT ON COLUMN ref.tr_age_age.age_code IS '1FW to 6FW and 1SW to 2SW';
COMMENT ON COLUMN ref.tr_age_age.age_description IS 'Description of the age';
COMMENT ON COLUMN ref.tr_age_age.age_definition IS 'Definition of the age';
COMMENT ON COLUMN ref.tr_age_age.age_icesvalue IS 'Code for the age in the ICES database';
COMMENT ON COLUMN ref.tr_age_age.age_icesguid IS 'GUID in the ICES database';
COMMENT ON COLUMN ref.tr_age_age.age_icestablesource IS 'Source table in ICES vocab';
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_age_age;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.27: Table of age

age_value	age_envir	age_code	age_description	age_definition
1	Freshwater	0FW	0 year in freshwater	Age of juvenile fish in their first y
1	Freshwater	1FW	1 year in freshwater	NA
2	Freshwater	2FW	2 years in freshwater	NA
3	Freshwater	3FW	3 years in freshwater	NA
4	Freshwater	4FW	4 years in freshwater	NA
5	Freshwater	5FW	5 years in freshwater	NA
6	Freshwater	6FW	6 years in freshwater	NA
1	Seawater	1SW	1 year in seawater	NA
2	Seawater	2SW	2 years in seawater	NA
NA	Seawater	MSW	Two years or more in seawater	NA
NA	Freshwater	1+	Older than one year in freshwater	Groups all fishes older than one y

ANSWER WGTRUTTA : Iain

Are the ages obtained from scale reading? In our Marine Directorate database we are careful to separate scale read ages from "guessed age" derived from sizes or field derived - observed (e.g. fry = 0+). If these are not clearly recorded in different areas of the database, is there somewhere to store information on the protocols? In MD database we store information on projects / campaigns (describe why and how data collected) and also protocols applied at SiteVisit level. The general definition looks OK. Although, when you store adults, how do you record more complex patterns e.g. 3SW with a spawning mark after

year 2? In MD database all this is recorded on the “Scale Record” that links to individual fish. > DIASPARA

2.20 Sex (tr_sex_sex)

There is a referential about sex in ICES (thanks Maria and Joana for pointing that out…) see Table Table 2.28.

```
TS_SEXCO <- icesVocab::getCodeList('SEXCO')
kable(TS_SEXCO, caption = "TS_SEXCO") |> kable_styling(bootstrap_options = c("striped", "hov
```

Ta

Id	Guid	Key	Description
26730	57201ebe-e901-41b9-9939-f54cadb83ebb	F	Female
26731	dba034ab-7e05-4332-8586-f1b33e4c2093	H	Hermaphrodite
26732	48dabe79-03f3-4fc2-ab66-cb1cc52cc735	I	Immature - attempt made but sex could not be d
26733	9d81d4b2-dfc7-4fb0-86ee-b66e13a2125c	M	Male
234003	484e5245-7f83-4d0d-957c-43f17c17f0af	T	Transitional
130028	ea6f732e-cef7-4f5a-abbc-71baa2f4dcfe	U	Undetermined - no attempt made
26734	4056362e-05ec-4f0f-8e24-d99e30821fe0	X	Mixed

SQL code to create table `tr_sex_sex`

```
DROP TABLE IF EXISTS ref.tr_sex_sex CASCADE;
CREATE TABLE ref.tr_sex_sex (
  sex_id SERIAL PRIMARY KEY,
  sex_code TEXT NOT NULL CONSTRAINT uk_sex_code UNIQUE,
  sex_description TEXT,
  sex_icesvalue character varying(4),
  sex_icesguid uuid,
  sex_icestablesource text
);

COMMENT ON TABLE ref.tr_sex_sex IS 'Table of possible sex values corresponding to the 7 scal
COMMENT ON COLUMN ref.tr_sex_sex.sex_id IS 'Integer, primary key of the table';
COMMENT ON COLUMN ref.tr_sex_sex.sex_code IS 'The code of sex';
COMMENT ON COLUMN ref.tr_sex_sex.sex_description IS 'Definition of the sex nature';
COMMENT ON COLUMN ref.tr_sex_sex.sex_icesvalue IS 'Code (Key) of the sex in ICES db';
COMMENT ON COLUMN ref.tr_sex_sex.sex_icesguid IS 'UUID (guid) of ICES, you can access by pas

GRANT ALL ON ref.tr_sex_sex TO diaspara_admin;
GRANT SELECT ON ref.tr_sex_sex TO diaspara_read;
```

```r
sex <- icesVocab::getCodeList('SEXCO')

sex <- sex |>
  rename("sex_icesguid"="GUID",  "sex_icesvalue" = "Key", "sex_description" = "Description")
  select ( sex_icesvalue, sex_icesguid, sex_description) |>
  mutate(sex_icestablesource = "SEXCO",
         sex_id = 1:7,
         sex_code = sex_icesvalue) |>
  select(sex_id, sex_code, sex_description, sex_icesvalue, sex_icesguid, sex_icestablesource

DBI::dbWriteTable(con_diaspara_admin, "temp_sex", sex, overwrite = TRUE)


DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_sex_sex
(sex_id, sex_code, sex_description, sex_icesvalue, sex_icesguid, sex_icestablesource)
SELECT sex_id, sex_code, sex_description, sex_icesvalue, sex_icesguid::uuid, sex_icestablesc
FROM temp_sex")# 4


DBI::dbExecute(con_diaspara_admin, "DROP table temp_sex")
```

```r
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_sex_sex;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.29: Table of sex

sex_id	sex_code	sex_description	sex_icesvalue	sex_ic
1	F	Female	F	57201
2	H	Hermaphrodite	H	dba03
3	I	Immature - attempt made but sex could not be destinguished	I	48dab
4	M	Male	M	9d81d
5	T	Transitional	T	484e5
6	U	Undetermined - no attempt made	U	ea6f73
7	X	Mixed	X	40563

2.21 Gear (ref.tr_gear_gea)

The gears dictionary is set by FAO and used in EU link. There is a gear dictionary in ICES but it's used to describe the type of engine on experimental trawling surveys. There might be a need to include more passive engine like trap and fishways.

SQL code to create table `tr_gear_gea`

```sql
DROP TABLE IF EXISTS ref.tr_gear_gea;
CREATE TABLE ref.tr_gear_gea (
  gea_id serial4 NOT NULL,
  gea_code text NOT NULL,
  gea_description text NULL,
  gea_icesvalue varchar(4) NULL,
  gea_icesguid uuid NULL,
  gea_icestablesource text NULL,
  CONSTRAINT tr_gear_gea_pkey PRIMARY KEY (gea_id),
  CONSTRAINT uk_gea_code UNIQUE (gea_code)
);
COMMENT ON TABLE ref.tr_gear_gea IS 'Table of fishing gears coming from FAO https://openknow
COMMENT ON COLUMN ref.tr_gear_gea.gea_id IS 'Id of the gear internal serial';
COMMENT ON COLUMN ref.tr_gear_gea.gea_issscfg_code IS 'Isssfg code of the gear';
COMMENT ON COLUMN ref.tr_gear_gea.gea_description IS 'English name of the gear';
COMMENT ON COLUMN ref.tr_maturity_mat.mat_icesvalue IS 'Code (Key) of the maturity in ICES c
COMMENT ON COLUMN ref.tr_maturity_mat.mat_icesguid IS 'UUID (guid) of ICES, you can access t
GRANT ALL ON ref.tr_gear_gea TO diaspara_admin;
GRANT SELECT ON ref.tr_gear_gea TO diaspara_read;




-- manual code to merge tr_gear_with ICES DB


UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='PS'
  WHERE gea_id=1;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='SB'
  WHERE gea_id=4;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='SBV'
  WHERE gea_id=5;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='TBB'
  WHERE gea_id=7;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='LLS'
  WHERE gea_id=45;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='LLD'
  WHERE gea_id=46;
```

```
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='LTL'
  WHERE gea_id=49;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='LX'
  WHERE gea_id=50;
UPDATE ref.tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='MIS'
  WHERE gea_id=59;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='HMD'
        WHERE gea_id=20;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='LA'
        WHERE gea_id=27;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='GND'
        WHERE gea_id=30;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='GNC'
        WHERE gea_id=31;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue=''
        WHERE gea_id=32;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='GRT'
        WHERE gea_id=33;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='GTN'
        WHERE gea_id=34;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='FPO'
        WHERE gea_id=37;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='FYK'
        WHERE gea_id=38;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType',gea_icesvalue='LHP'
        WHERE gea_id=43;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=1;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=4;
```

```
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=5;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=7;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=45;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=46;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=49;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=50;
UPDATE "ref".tr_gear_gea
        SET gea_icestablesource='GearType'
        WHERE gea_id=59; --28
UPDATE "ref".tr_gear_gea
  SET gea_icestablesource='GearType'
  WHERE gea_id=7;
UPDATE "ref".tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='OTB'
  WHERE gea_id=8;
UPDATE "ref".tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='OTT'
  WHERE gea_id=10;
UPDATE "ref".tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='PTB'
  WHERE gea_id=11;
UPDATE "ref".tr_gear_gea
  SET gea_icestablesource='GearType',gea_icesvalue='OTM'
  WHERE gea_id=13;


ALTER TABLE tr_gear_gea SET gea_code
```

```
gea <- dbGetQuery(con_wgeel_distant, "SELECT * FROM ref.tr_gear_gea")

gea <- gea |>
  rename("gea_code"="gea_issscfg_code",  "gea_description" = "gea_name_en") |>
  select(-gea_id) |>
```

```
  arrange(gea_code) |>
  mutate(gea_icestablesource = NA,
         gea_icesvalue = NA,
         gea_icesguid = as.character(NA),
         gea_id = 1:nrow(gea)
  ) |>
  select(gea_id, gea_code, gea_description, gea_icesvalue, gea_icesguid, gea_icestablesource

DBI::dbWriteTable(con_diaspara_admin, "temp_gear", gea, overwrite = TRUE)


DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_gear_gea
(gea_id, gea_code, gea_description, gea_icesvalue, gea_icesguid, gea_icestablesource)
SELECT gea_id, gea_code, gea_description, gea_icesvalue, gea_icesguid::uuid, gea_icestables
FROM temp_gear")# 60


DBI::dbExecute(con_diaspara_admin, "DROP table temp_gear")
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_gear_gea;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.30: Table of gears

gea_id	gea_code	gea_description	gea_icesvalue
2	01.2	Surrounding nets without purse lines	NA
3	01.9	Surrounding nets (nei)	NA
6	02.9	Seine nets (nei)	NA
9	03.13	Twin bottom otter trawls	NA
12	03.19	Bottom trawls (nei)	NA
14	03.22	Midwater pair trawls	NA
15	03.29	Midwater trawls (nei)	NA
16	03.3	Semipelagic trawls	NA
17	03.9	Trawls (nei)	NA
18	04.1	Towed dredges	NA
19	04.2	Hand dredges	NA
21	04.9	Dredges (nei)	NA
22	05.1	Portable lift nets	NA
23	05.2	Boat-operated lift nets	NA
24	05.3	Shore-operated stationary lift nets	NA
25	05.9	Lift nets (nei)	NA
26	06.1	Cast nets	NA

28	06.9	Falling gear (nei)	NA
29	07.1	Set gillnets (anchored)	NA
35	07.9	Gillnets and entangling nets (nei)	NA
36	08.1	Stationary uncovered pound nets	NA
39	08.4	Stow nets	NA
40	08.5	Barriers, fences, weirs, etc.	NA
41	08.6	Aerial traps	NA
42	08.9	Traps (nei)	NA
44	09.2	Mechanized lines and pole-and-lines	NA
47	09.39	Longlines (nei)	NA
48	09.4	Vertical lines	NA
51	10.1	Harpoons	NA
52	10.2	Hand Implements (Wrenching gear, Clamps, Tongs, Rakes, Spears)	NA
53	10.3	Pumps	NA
54	10.4	Electric fishing	NA
55	10.5	Pushnets	NA
56	10.6	Scoopnets	NA
57	10.7	Drive-in nets	NA
58	10.8	Diving	NA
20	04.3	Mechanized dredges	HMD
27	06.2	Cover pots/Lantern nets	LA
30	07.2	Drift gillnets	GND
31	07.3	Encircling gillnets	GNC
32	07.4	Fixed gillnets (on stakes)	
33	07.5	Trammel nets	GRT
34	07.6	Combined gillnets-trammel nets	GTN
37	08.2	Pots	FPO
38	08.3	Fyke nets	FYK
43	09.1	Handlines and hand-operated pole-and-lines	LHP
1	01.1	Purse seines	PS
4	02.1	Beach seines	SB
5	02.2	Boat seines	SBV
45	09.31	Set longlines	LLS
46	09.32	Drifting longlines	LLD
49	09.5	Trolling lines	LTL
50	09.9	Hooks and lines (nei)	LX
59	10.9	Gear nei	MIS
7	03.11	Beam trawls	TBB
8	03.12	Single boat bottom otter trawls	OTB
10	03.14	Multiple bottom otter trawls	OTT
11	03.15	Bottom pair trawls	PTB
13	03.21	Single boat midwater otter trawls	OTM

60	99.9	Gear not known

Other dictionaries in ICES

The geartype which corresponds to metier 4 is sourced by the DCF and maintained by the JRC (contact can be provided by ICES). The Sampler type SMTYP provides a dictionary of the scientific gear used in monitoring, this one can be updated

Other source of definition (if needed)

Method used to monitor eel in the mediterranean are referenced in detail in this report.

2.22 ICES areas

We have create entries in the table 'tr_fishingarea_fia for FAO major fishing area (27, 21, 37, 34, 31).

- 27 Atlantic, Northeast
- 21 Atlantic, Northwest
- 37 Mediterranean and Black Sea
- 34 Atlantic Eastern Central
- 31 Atlantic, Western Central

source : GFCM geographical subareas https://www.fao.org/gfcm/data/maps/gsas/fr/ https://gfcmsitestorage.blob.core.windows.net/website/5.Data/ArcGIS/GSAs_simplified_updated_division%20(2).zip

source : NAFO divisions https://www.nafo.int/Data/GIS https://www.nafo.int/Portals/0/GIS/Divisions.zip

source : ICES statistical areas https://gis.ices.dk/shapefiles/ICES_areas.zip https://gis.ices.dk/geonetwork/srv/eng/catalog.search#/metadata/c784a0a3-752f-4b50-b02f-f225f6c815eb

The rest of the world was found on a computer. Cannot trace the source, it's exaclty the same for NAFO but changed in the Med and ICES. For some reasons was not complete in table from wgeel so have to download it again to postgres.

Values for geom have been updated from ICES areas, the new boundaries are different, however, there are more than the previous ones. The values for Areas and Subareas have not been updated but these are for wide maps so we'll leave it as it is.

```
dbExecute(con_diaspara_admin, "DROP TABLE IF EXISTS ref.tr_fishingarea_fia
CASCADE;")
dbExecute(con_diaspara_admin,
          "
  CREATE TABLE ref.tr_fishingarea_fia
  (
    fia_level TEXT,
    fia_code TEXT,
    fia_status numeric,
    fia_ocean TEXT,
    fia_subocean TEXT,
    fia_area TEXT,
    fia_subarea TEXT,
    fia_division TEXT,
    fia_subdivision TEXT,
    fia_unit TEXT,
    fia_name TEXT NULL,
    geom geometry(MultiPolygon,4326),
    CONSTRAINT tr_fishingarea_fia_pkey PRIMARY KEY (fia_code),
    CONSTRAINT uk_fia_subdivision UNIQUE (fia_unit)
  )
  ;
")


# start with initial FAO dataset

#area_all <- dbGetQuery(con_diaspara_admin, "SELECT * FROM area.\"FAO_AREAS\"
# WHERE f_area IN ('21','27','31','34','37') ;")

# In this table all geom are mixed from unit to division.
# It only make sense to extract for a unique f_level


# TODO add species, wk
dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_fishingarea_fia
SELECT
   initcap(f_level) AS fia_level,
   f_code AS  fia_code,
   f_status AS  fia_status,
   ocean AS  fia_ocean,
   subocean AS  fia_subocean,
   f_area AS  fia_area,
   f_subarea AS  fia_subarea,
   f_division AS  fia_division,
```

```
   f_subdivis AS  fia_subdivision,
   f_subunit AS  fia_unit,
   NULL as fia_name,
   geom
  FROM area.\"FAO_AREAS\"
  WHERE f_area IN ('21','27','31','34','37')
") # 187

# Replace values ices
dbExecute(con_diaspara_admin, "UPDATE ref.tr_fishingarea_fia
    set geom = st_transform(are.geom, 4326)
    FROM
    area.\"ICES_Areas_20160601_cut_dense_3857\" are
    WHERE area_full = fia_code;") # 66
# Replace values NAFO (nothing to do ...)



# Replace values GFCM
dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_fishingarea_fia
SELECT
'Subdivision' AS fia_level,
f_gsa AS fia_code,
1 AS fia_status,
'Atlantic' AS fia_ocean,
3 AS fia_subocean,
f_area,
f_subarea,
f_division,
f_gsa AS fia_subdivision,
NULL AS fia_unit,
smu_name AS fia_name,
geom
FROM area.\"GSAs_simplified_division\";") # 32

dbExecute(con_diaspara_admin, "GRANT ALL ON ref.tr_fishingarea_fia
          TO diaspara_admin;")
dbExecute(con_diaspara_admin, "GRANT SELECT ON ref.tr_fishingarea_fia
          TO diaspara_read;")

dbExecute(con_diaspara_admin, "COMMENT ON TABLE ref.tr_fishingarea_fia
IS 'Table of fishing areas, attention, different levels of geometry
details are present in the table, area, subarea, division, subdivision, unit,
most query will use WHERE
 fia_level = ''Subdivision''';")
```

```r
library(rnaturalearth)
world <- ne_countries(scale = "small", returnclass = "sf")


if (file.exists("data/fishingareas_major.Rdata"))
  load("data/fishingareas_major.Rdata") else {
    fishing_areas_major <- sf::st_read(con_diaspara,
                                       query = "SELECT fia_code, ST_MakeValid(geom)
                           from ref.tr_fishingarea_fia
                           WHERE fia_level = 'Major'") |>
      sf::st_transform(4326)
    save(fishing_areas_major, file="data/fishing_areas_major.Rdata")
  }
load("data/country_sf.Rdata")
crs_string <- "+proj=ortho +lon_0=-30 +lat_0=30"

ocean <- sf::st_point(x = c(0,0)) |>
  sf::st_buffer(dist = 6371000) |>
  sf::st_sfc(crs = crs_string)
area_sf2 <-  fishing_areas_major |>
  sf::st_intersection(ocean |> sf::st_transform(4326)) |>
  sf::st_transform(crs = crs_string)

country_sf2 <-  country_sf |>
  sf::st_intersection(ocean |> sf::st_transform(4326)) |>
  sf::st_transform(crs = crs_string)



g <- ggplot() +
  geom_sf(data = ocean, fill = "deepskyblue4", color = NA) +
  geom_sf(data = area_sf2, aes(fill = fia_code), color="white", lwd = .1) +
  geom_sf(data = world, fill="black", color="grey20") +
  geom_sf(data= country_sf2 , fill= "grey10",color="grey30")  +
  theme_void()
#scale_fill_discrete(guide = "none") +


# this part is used to avoid long computations
png(filename="images/fig-fishingareas_major.png",width=600, height=600, res=300, bg="transpa
print(g)
dev.off()
```

```r
if (file.exists("data/fishingareas_subarea.Rdata")) load("data/fishingareas_subarea.Rdata")
  fishing_areas_subarea <- sf::st_read(con_diaspara,
```

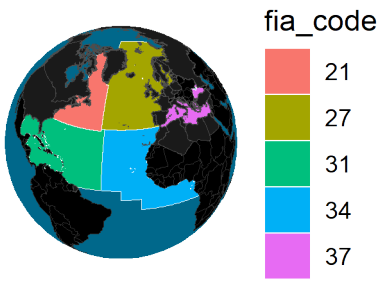


Figure 2.14: Map of ICES fishing areas at Major level, source NAFO, FAO, ICES, GFCM.

```
                                        query = "SELECT fia_code, ST_MakeValid(geom)
                          from ref.tr_fishingarea_fia
                          WHERE fia_level = 'Subarea'") |>
    sf::st_transform(4326)
  save(fishing_areas_subarea, file="data/fishing_areas_subarea.Rdata")
}
crs_string <- "+proj=ortho +lon_0=-30 +lat_0=30"

area_sf2 <-  fishing_areas_subarea |>
  sf::st_intersection(ocean |> sf::st_transform(4326)) |>
  sf::st_transform(crs = crs_string) # reproject to ortho

g <- ggplot() +
  geom_sf(data = ocean, fill = "deepskyblue4", color = NA) +
  geom_sf(data = area_sf2, aes(fill = fia_code), color="white", lwd = .1) +
  geom_sf(data = world, fill="black", color="grey20") +
  geom_sf(data= country_sf2 , fill= "grey10",color="grey30")  +
  scale_fill_discrete(guide = "none")  +
  theme_void()

# this part is used to avoid long computations
png(filename="images/fig-fishingareas_subarea.png", bg="transparent")
print(g)
dev.off()
```

```
if (file.exists("data/fishingareas_division.Rdata")) load("data/fishingareas_division.Rdata'
  fishing_areas_division <- sf::st_read(con_diaspara,
                                        query = "SELECT fia_code, ST_MakeValid(geom)
                          from ref.tr_fishingarea_fia
```

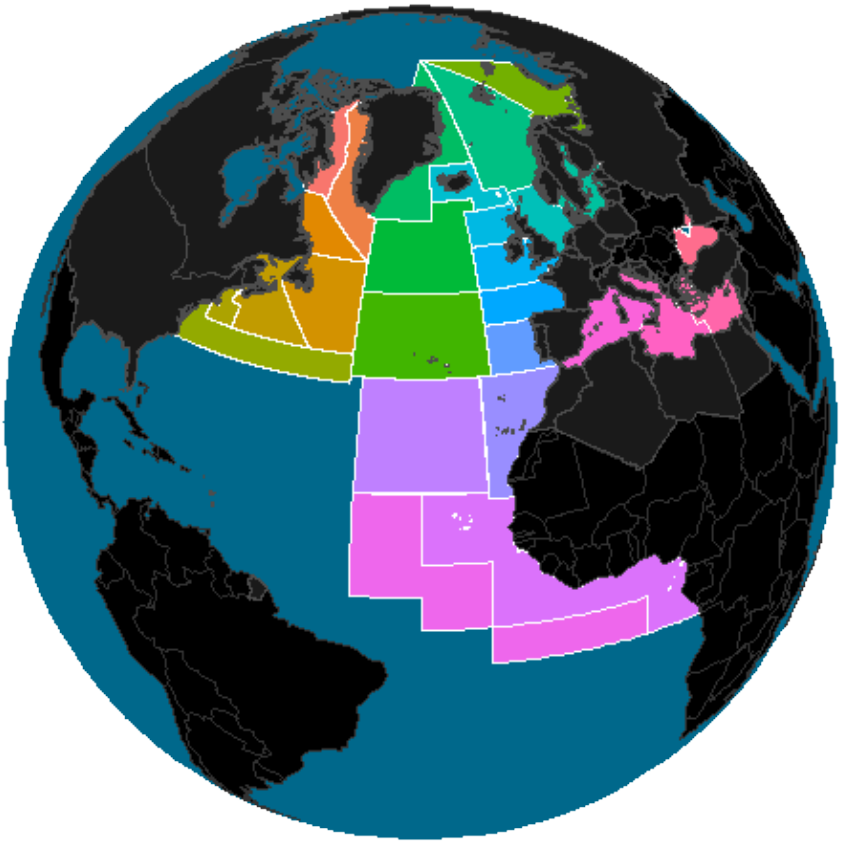


Figure 2.15: Map of ICES fishing areas at Subarea level, source NAFO, FAO, ICES, GFCM.

```
                          WHERE fia_level = 'Division'") |>
    sf::st_transform(4326)
  save(fishing_areas_division, file="data/fishing_areas_division.Rdata")
}

crs_string <- "+proj=ortho +lon_0=-30 +lat_0=30"


ocean <- sf::st_point(x = c(0,0)) |>
  sf::st_buffer(dist = 6371000) |>
  sf::st_sfc(crs = crs_string)

area_sf2 <-  fishing_areas_division |>
  sf::st_intersection(ocean |> sf::st_transform(4326)) |>
  sf::st_transform(crs = crs_string)

g <- ggplot() +
  geom_sf(data = ocean, fill = "deepskyblue4", color = NA) +
  geom_sf(data = area_sf2, aes(fill = fia_code), color="white", lwd = .1) +
  geom_sf(data = world, fill="black", color="grey20") +
  geom_sf(data= country_sf2 , fill= "grey10",color="grey30")  +
  scale_fill_discrete(guide = "none") +
  theme_void()

png(filename="images/fig-fishingareas_division.png", bg="transparent")
print(g)
dev.off()
```

```
if (file.exists("data/fishingareas_subdivision.Rdata")) load("data/fishingareas_subdivision.
  fishing_areas_subdivision <- sf::st_read(con_diaspara,
                                           query = "SELECT fia_code, ST_MakeValid(geom)
                          from ref.tr_fishingarea_fia
                          WHERE fia_level = 'Subdivision'") |>
    sf::st_transform(4326)
  save(fishing_areas_subdivision, file="data/fishing_areas_subdivision.Rdata")
}

crs_string <- "+proj=ortho +lon_0=-30 +lat_0=30"


ocean <- sf::st_point(x = c(0,0)) |>
  sf::st_buffer(dist = 6371000) |>
  sf::st_sfc(crs = crs_string)

area_sf2 <-  fishing_areas_subdivision |>
```

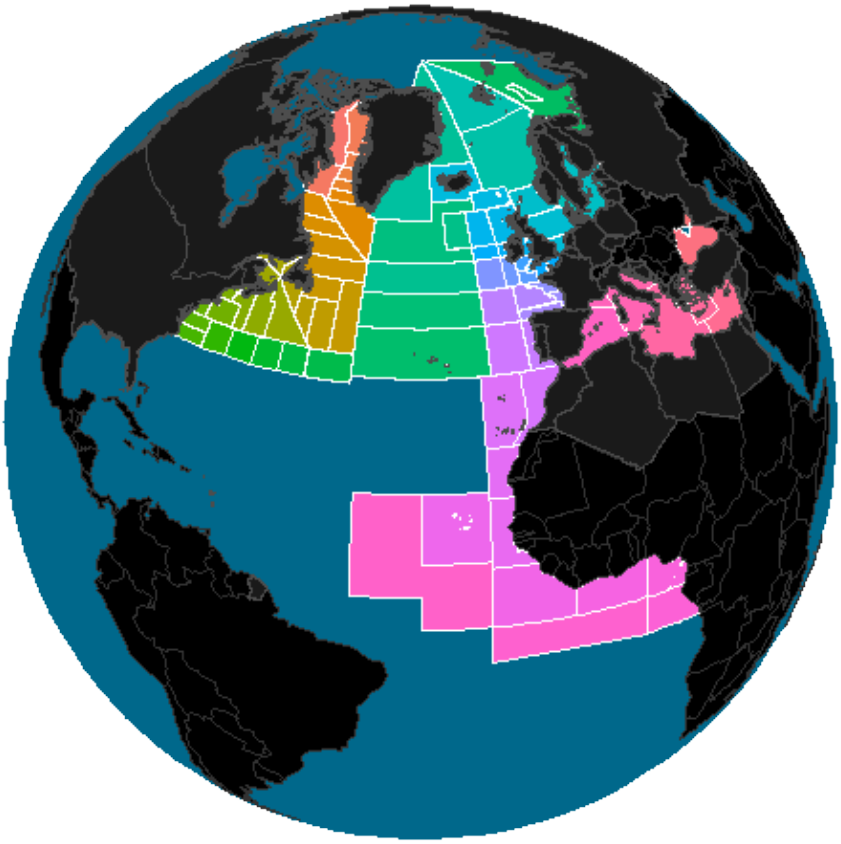


Figure 2.16: Map of ICES fishing areas at division level, source NAFO, FAO, ICES, GFCM.

```
  sf::st_intersection(ocean |> sf::st_transform(4326)) |>
  sf::st_transform(crs = crs_string)

g <- ggplot() +
  geom_sf(data = ocean, fill = "deepskyblue4", color = NA) +
  geom_sf(data = area_sf2, aes(fill = fia_code), color= "white", lwd = .1) +
  geom_sf(data = world, fill="black", color="grey20") +
  geom_sf(data= country_sf2 , fill= "grey10",color="grey30")  +
  scale_fill_discrete(guide = "none") +
  theme_void()

png(filename="images/fig-fishingareas_subdivision.png", bg="transparent")
print(g)
dev.off()
```

2.23 Time period (ref.tr_timeperiod_tip)

WGBAST reports data per month, quarter, or half year. There is a vocabulary for quarters, couldn't find a vocab for half of year.

SQL code to create table `tr_timeperiod_tip`

```
-- maturity table code


DROP TABLE IF EXISTS ref.tr_timeperiod_tip CASCADE;
CREATE TABLE  ref.tr_timeperiod_tip (
  tip_id SERIAL PRIMARY KEY,
  tip_code TEXT NOT NULL CONSTRAINT uk_tip_code UNIQUE,
  tip_description TEXT,
  tip_icesvalue TEXT,
  tip_icesguid uuid,
  tip_icestablesource text
);

COMMENT ON TABLE ref.tr_timeperiod_tip IS 'Table of time periods';
COMMENT ON COLUMN ref.tr_timeperiod_tip.tip_id IS 'Integer, primary key of the table';
COMMENT ON COLUMN ref.tr_timeperiod_tip.tip_code IS 'The code of time period';
COMMENT ON COLUMN ref.tr_timeperiod_tip.tip_description IS 'Definition of the time period';
COMMENT ON COLUMN ref.tr_timeperiod_tip.tip_icesvalue IS 'Code (Key) of the time period in I
COMMENT ON COLUMN ref.tr_timeperiod_tip.tip_icesguid IS 'UUID (guid) of ICES, you can access
COMMENT ON COLUMN ref.tr_timeperiod_tip.tip_icestablesource IS 'Source table in ICES';
GRANT ALL ON ref.tr_timeperiod_tip TO diaspara_admin;
GRANT SELECT ON ref.tr_timeperiod_tip TO diaspara_read;
```

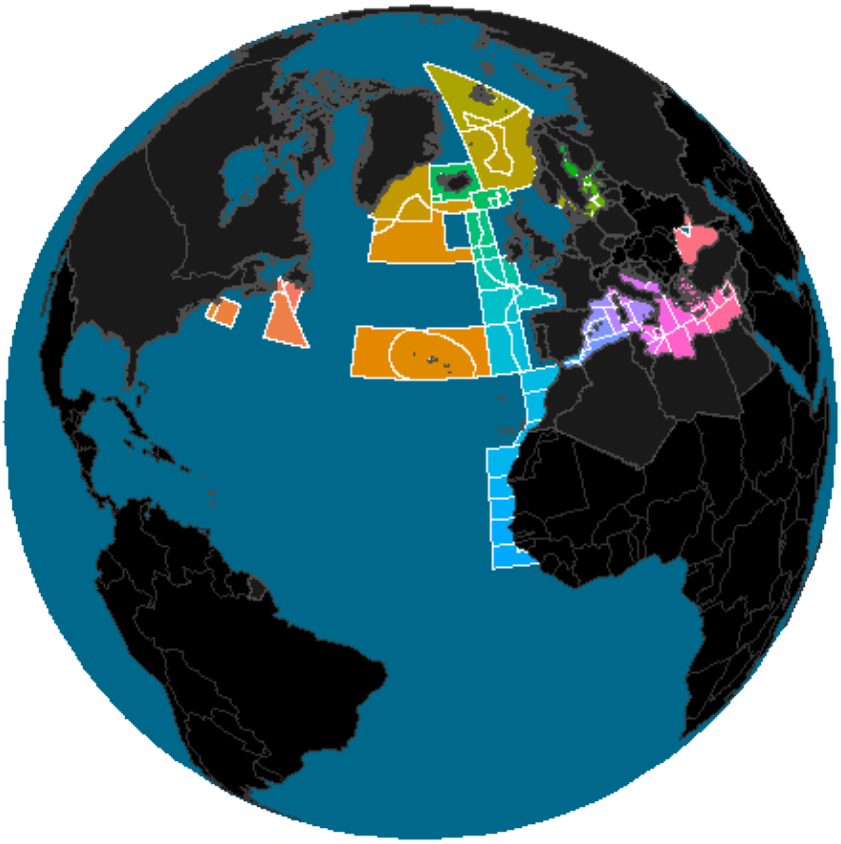


Figure 2.17: Map of ICES fishing areas at subdivision level, source NAFO, FAO, ICES, GFCM.

```
tp <- icesVocab::getCodeList('IC_SeasonType')
#I don't want this I want a code
tip <- data.frame(tip_id=1:4,
tip_code = c(tp$Key, "Half of Year"),
tip_description = c("Monthly data, from 1 to 12",  "Quarterly data from 1 to 4", "Year value
 tip_icesvalue = c(tp$Key,NA),
 tip_icesguid = c(tp$Guid,NA),
 tip_icestablesource =c(rep("IC_SeasonType", 3), NA))|>
  select(tip_id, tip_code, tip_description, tip_icesvalue, tip_icesguid, tip_icestablesource
tip$tip_icesguid <- as.character(tip$tip_icesguid)
DBI::dbWriteTable(con_diaspara_admin, "temp_tipr", tip, overwrite = TRUE)
DBI::dbExecute(con_diaspara_admin, "DELETE FROM ref.tr_timeperiod_tip")
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_timeperiod_tip
(tip_id, tip_code, tip_description, tip_icesvalue, tip_icesguid, tip_icestablesource)
SELECT tip_id, tip_code, tip_description, tip_icesvalue, tip_icesguid::uuid, tip_icestables
FROM temp_tipr")# 4
DBI::dbExecute(con_diaspara_admin, "DROP table temp_tipr")
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_timeperiod_tip;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.31: Table of gears

tip_id	tip_code	tip_description
1	Month	Monthly data, from 1 to 12
2	Quarter	Quarterly data from 1 to 4
3	Year	Year value of timeperiod should be NULL and year column filled
4	Half of Year	Half of year, either from 1 to 6 (included)=1, or from month 7 to 12 (included)=2

Only in WGBAST

Currently the time period are not used in WGEEL and WGNAS (where time period is always year), the referential is only used in WGBAST.

2.24 Data source (ref.tr_datasource_dts)

The source of fishery data is included in WGBAST see: WBAST description. The vocab described here is deprecated and superseeded by datasource.

During exchange with Data Centre (Thanks Maria for hinting at this change), the current way of handling estimated data, is to separate the source of data and the type of estimation. This is how it works currently in RDBES. So a catch

can coming from logbook, it is estimated, and when estimated a method must be provided if estimated (estimation method).

SQL code to create table `tr_datasource_dts`

```
-- Table for quality

DROP TABLE IF EXISTS ref.tr_datasource_dts CASCADE;
CREATE  TABLE ref.tr_datasource_dts (
  dts_id int4 PRIMARY KEY,
  dts_code TEXT NOT NULL CONSTRAINT uk_dts_code UNIQUE,
  dts_description text NULL,
  dts_icesvalue TEXT,
  dts_icesguid uuid,
  dts_icestablesource text
);
COMMENT ON TABLE ref.tr_datasource_dts IS 'Table of data source values, e.g. logbooks, Exper
COMMENT ON COLUMN ref.tr_datasource_dts.dts_code IS 'Data srouce code';
COMMENT ON COLUMN ref.tr_datasource_dts.dts_description IS 'Data source description';
COMMENT ON COLUMN ref.tr_datasource_dts.dts_icesvalue IS 'Code (Key) of the time period in I
COMMENT ON COLUMN ref.tr_datasource_dts.dts_icesguid IS 'UUID (guid) of ICES, you can access
COMMENT ON COLUMN ref.tr_datasource_dts.dts_icestablesource IS 'Source table in ICES';

GRANT ALL ON ref.tr_datasource_dts TO diaspara_admin;
GRANT SELECT ON ref.tr_datasource_dts TO diaspara_read;
```

```
dts <- icesVocab::getCodeList('DataSource')
#
dts <- dts |>
  rename("dts_icesguid"="Guid",  "dts_code" = "Key", "dts_description" = "Description") |>
  select ( dts_code, dts_icesguid, dts_description)  |>
  mutate(dts_icestablesource = "DataSource",
         dts_id = 1:nrow(dts),
         dts_icesvalue = dts_code) |>
  select(dts_id, dts_code, dts_description, dts_icesvalue, dts_icesguid, dts_icestablesource



DBI::dbWriteTable(con_diaspara_admin, "temp_dtsr", dts, overwrite = TRUE)
DBI::dbExecute(con_diaspara_admin, "DELETE FROM ref.tr_datasource_dts")
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_datasource_dts
(dts_id, dts_code, dts_description, dts_icesvalue, dts_icesguid, dts_icestablesource)
SELECT dts_id, dts_code, dts_description, dts_icesvalue, dts_icesguid::uuid, dts_icestablesc
FROM temp_dtsr")# 14
DBI::dbExecute(con_diaspara_admin, "DROP table temp_dtsr")
```

```
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_datasource_dts (dts_id,dts_code,dts_c
    VALUES (15,'Smolt','Smolt count');")
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_datasource_dts (dts_id,dts_code,dts_c
    VALUES (16,'Parr','Parr densities (electrofishing)');")
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_datasource_dts (dts_id,dts_code,dts_c
    VALUES (17,'Spawner','Spawner count');")
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_datasource_dts (dts_id,dts_code,dts_c
    VALUES (18,'Stocking','Stocking data');")
```

```
dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_datasource_dts;") |>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

Table 2.32: Tab
fit in there but
no correspondа
The smolt esti

dts_id	dts_code	dts_description
1	CombOD	Combination of official data sources (logbooks, sales notes, other forms)
2	Crew	Vessel Crew. Data obtained directly from the crew of the vessel (not via any offic
3	Exprt	Expert knowledge. Data is estimated using expert knowledge about the activity.
4	HarbLoc	Harbor location (Landing port harbor used as basis to estimate fishing geographic
5	Logb	Logbook data
6	NotApplicable	Not applicable
7	Observer	Observer. Data obtained directly by an observer.
8	OthDF	Other declarative forms (i.e. landing declarations and national declarative forms)
9	PosDat	Positional data (other than VMS)
10	SalN	Sales notes
11	SampDC	Commercial sampling data (sampling methodologies specific to each country). Th
12	SampDS	Survey sampling data (sampling methodologies specific to each country)
13	Unknown	Not known
14	VMS	VMS data
15	Smolt	Smolt count
16	Parr	Parr densities (electrofishing)
17	Spawner	Spawner count
18	Stocking	Stocking data

Only in WGBAST

Currently the data source are only for WGBAST

2.25 Data basis (ref.tr_databasis_dtb)

SQL code to create table `tr_databasis_dtb`

```
-- Table of estimation methods when databasis is Estimated

DROP TABLE IF EXISTS ref.tr_databasis_dtb CASCADE;
CREATE  TABLE ref.tr_databasis_dtb (
  dtb_id int4 PRIMARY KEY,
  dtb_code TEXT NOT NULL CONSTRAINT uk_dtb_code UNIQUE,
  dtb_description text NULL,
  dtb_icesvalue TEXT,
  dtb_icesguid uuid,
  dtb_icestablesource text
);
COMMENT ON TABLE ref.tr_databasis_dtb IS 'Table of data basis';
COMMENT ON COLUMN ref.tr_databasis_dtb.dtb_code IS 'Data basis  code';
COMMENT ON COLUMN ref.tr_databasis_dtb.dtb_description IS 'Data basis description';
COMMENT ON COLUMN ref.tr_databasis_dtb.dtb_icesvalue IS 'Code (Key) of the Data basis in ICE
COMMENT ON COLUMN ref.tr_databasis_dtb.dtb_icesguid IS 'UUID (guid) of ICES ';
COMMENT ON COLUMN ref.tr_databasis_dtb.dtb_icestablesource IS 'Source table in ICES';

GRANT ALL ON ref.tr_databasis_dtb TO diaspara_admin;
GRANT SELECT ON ref.tr_databasis_dtb TO diaspara_read;
```

```
dtb <- icesVocab::getCodeList('DataBasis')
#
dtb <- dtb |>
  rename("dtb_icesguid"="Guid",  "dtb_code" = "Key", "dtb_description" = "Description") |>
  select ( dtb_code, dtb_icesguid, dtb_description)  |>
  mutate(dtb_icestablesource = "DataBasis",
         dtb_id = 1:nrow(dtb),
         dtb_icesvalue = dtb_code) |>
  select(dtb_id, dtb_code, dtb_description, dtb_icesvalue, dtb_icesguid, dtb_icestablesource



DBI::dbWriteTable(con_diaspara_admin, "temp_dtbr", dtb, overwrite = TRUE)
DBI::dbExecute(con_diaspara_admin, "INSERT INTO ref.tr_databasis_dtb
(dtb_id, dtb_code, dtb_description, dtb_icesvalue, dtb_icesguid, dtb_icestablesource)
SELECT dtb_id, dtb_code, dtb_description, dtb_icesvalue, dtb_icesguid::uuid, dtb_icestables
FROM temp_dtbr")# 5
DBI::dbExecute(con_diaspara_admin, "DROP table temp_dtbr")
```

2.26 Data estimation method ref.tr_estimation-method_esm

This table will be inherited, we are proposing working group specific estimation methods. Typically estimation methods of WGBAST. These will only be used if sto_dtb_code = 'Estimated'

1. Complete count of smolts.
2. Sampling of smolts and estimate of total smolt run size.
3. Estimate of smolt run from parr production by relation developed in the same river.
4. Estimate of smolt run from parr production by relation developed in another river.
5. Inference of smolt production from data derived from similar rivers in the region.
6. Count of spawners.
7. Estimate inferred from stocking of reared fish in the river.
8. Salmon catch, exploitation and survival estimate.

SQL code to create table `ref.tr_estimationmethod_esm`

```
-- Table of estimation methods when databasis is Estimated

DROP TABLE IF EXISTS ref.tr_estimationmethod_esm CASCADE;
CREATE  TABLE ref.tr_estimationmethod_esm (
  esm_id int4 PRIMARY KEY,
  esm_code TEXT NOT NULL CONSTRAINT uk_esm_code UNIQUE,
  esm_description text NULL,
  esm_wkg_code TEXT NOT NULL,
CONSTRAINT fk_esm_wkg_code  FOREIGN KEY (esm_wkg_code)
REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
ON UPDATE CASCADE ON DELETE RESTRICT,
  esm_icesvalue TEXT,
  esm_icesguid uuid,
  esm_icestablesource text
);
COMMENT ON TABLE ref.tr_estimationmethod_esm IS 'Table of table estimation method, provided
COMMENT ON COLUMN ref.tr_estimationmethod_esm.esm_code IS 'Estimation method code';
COMMENT ON COLUMN ref.tr_estimationmethod_esm.esm_description IS 'Estimation method  descrip
COMMENT ON COLUMN ref.tr_estimationmethod_esm.esm_icesvalue IS 'Code (Key) of the Estimation
COMMENT ON COLUMN ref.tr_estimationmethod_esm.esm_icesguid IS 'UUID (guid) of ICES ';
COMMENT ON COLUMN ref.tr_estimationmethod_esm.esm_icestablesource IS 'Source table in ICES';
COMMENT ON COLUMN ref.tr_estimationmethod_esm.esm_wkg_code
IS 'Code of the working group, one of WGBAST, WGEEL, WGNAS, WKTRUTTA';
GRANT ALL ON ref.tr_estimationmethod_esm TO diaspara_admin;
GRANT SELECT ON ref.tr_estimationmethod_esm TO diaspara_read;
```

Only in WGBAST

Currently the estimation method are only for WGBAST

Note

The estimation method are inherited, so there is a common table and estimation methods are created in daughter tables in each working group, they are inherited. See Section 6.2

Chapter 3

Metadata

3.1 Metadata (dat.t_metadata_met)

The code for creating metadata is listed below

SQL code to create table `dat.t_metadata_met`

```
DROP TABLE IF EXISTS dat.t_metadata_met CASCADE;
CREATE TABLE dat.t_metadata_met (
  met_var TEXT NOT NULL,
  met_spe_code character varying(3) NOT  NULL,
  met_wkg_code TEXT NOT NULL,
  met_ver_code TEXT NULL,
  met_oty_code TEXT NOT NULL,
  met_nim_code TEXT NOT NULL,
  met_dim integer ARRAY,
  met_dimname TEXT ARRAY,
  met_modelstage TEXT NULL,
  met_type TEXT NULL,
  met_location TEXT NULL,
  met_fishery TEXT NULL,
  met_mtr_code TEXT NULL,
  met_des_code TEXT NULL,
  met_uni_code TEXT NULL,
  met_cat_code TEXT NULL,
  met_definition TEXT NULL,
  met_deprecated BOOLEAN DEFAULT FALSE,
  CONSTRAINT t_metadata_met_pkey PRIMARY KEY(met_var, met_spe_code),
  CONSTRAINT fk_met_spe_code FOREIGN KEY (met_spe_code)
  REFERENCES ref.tr_species_spe(spe_code)
```

```
  ON DELETE CASCADE
  ON UPDATE CASCADE,
    CONSTRAINT fk_met_wkg_code FOREIGN KEY (met_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_ver_code FOREIGN KEY (met_ver_code)
  REFERENCES ref.tr_version_ver(ver_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_oty_code FOREIGN KEY (met_oty_code)
  REFERENCES ref.tr_objecttype_oty (oty_code) ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_nim_code FOREIGN KEY (met_nim_code)
  REFERENCES ref.tr_nimble_nim (nim_code) ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_mtr_code FOREIGN KEY (met_mtr_code)
  REFERENCES ref.tr_metric_mtr(mtr_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_uni_code FOREIGN KEY (met_uni_code)
  REFERENCES ref.tr_units_uni(uni_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_cat_code FOREIGN KEY (met_cat_code)
  REFERENCES ref.tr_category_cat(cat_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE,
  CONSTRAINT fk_met_des_code FOREIGN KEY (met_des_code)
  REFERENCES ref.tr_destination_des(des_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE
);
COMMENT ON TABLE dat.t_metadata_met IS
'Table (metadata) of each variable (parameter) in the database.';
COMMENT ON COLUMN dat.t_metadata_met.met_var
IS 'Variable code, primary key on both met_spe_code and met_var.';
COMMENT ON COLUMN dat.t_metadata_met.met_spe_code
IS 'Species, ''127186'' (Salmo salar), ''127187'' (Salmo trutta), ''126281'' (Anguilla angui
COMMENT ON COLUMN dat.t_metadata_met.met_ver_code
IS 'Code on the version of the model, see table tr_version_ver.';
COMMENT ON COLUMN dat.t_metadata_met.met_oty_code
IS 'Object type, single_value, vector, matrix see table tr_objecttype_oty.';
COMMENT ON COLUMN dat.t_metadata_met.met_nim_code
IS 'Nimble type, one of data, constant, output, other.';
```

```
COMMENT ON COLUMN dat.t_metadata_met.met_dim
IS 'Dimension of the Nimble variable, use {10, 100, 100}
to insert the description of an array(10,100,100).';
COMMENT ON COLUMN dat.t_metadata_met.met_dimname
IS 'Dimension of the variable in Nimble, use {''year'', ''stage'', ''area''}.';
COMMENT ON COLUMN dat.t_metadata_met.met_modelstage
IS 'Currently one of fit, other, First year.';
COMMENT ON COLUMN dat.t_metadata_met.met_type
IS 'Type of data in the variable, homewatercatches, InitialISation first year,
abundance ....';
COMMENT ON COLUMN dat.t_metadata_met.met_location
IS 'Describe process at sea, e.g. Btw. FAR - GLD fisheries, or Aft. Gld fISheries.';
COMMENT ON COLUMN dat.t_metadata_met.met_fishery
IS 'Description of the fishery.';
COMMENT ON COLUMN dat.t_metadata_met.met_des_code
IS 'Outcome of the fish, e.g. Released (alive), Seal damage,
Removed (from the environment), references table tr_destination_des.';
COMMENT ON COLUMN dat.t_metadata_met.met_uni_code
IS 'Unit, references table tr_unit_uni.';
COMMENT ON COLUMN dat.t_metadata_met.met_cat_code
IS 'Broad category of data or parameter,
catch, effort, biomass, mortality, count ...references table tr_category_cat.';
COMMENT ON COLUMN dat.t_metadata_met.met_mtr_code
IS 'Code of the metric, references tr_metric_mtr, Estimate, Bound, SD, CV ....';
COMMENT ON COLUMN dat.t_metadata_met.met_definition
IS 'Definition of the metric.';
COMMENT ON COLUMN dat.t_metadata_met.met_deprecated
IS'Is the variable still used ?';

GRANT ALL ON dat.t_metadata_met TO diaspara_admin;
GRANT SELECT ON dat.t_metadata_met TO diaspara_read;




/*
SELECT * FROM refsalmoglob."database" JOIN
datnas.t_metadata_met AS tmm ON tmm.met_var = var_mod
WHERE "year" IS NULL
*/
```

met_var	met_spe_code	met_wkg_code	met_ver_code	met_oty_code	r
log_C5_NAC_1_lbnf_sd	127186	WGNAS	WGNAS-2024-1	Vector	l

log_C5_NAC_2_lbnf_lab_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C5_NAC_2_lbnf_oth_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C5_NEC_1_far_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_2_gld_tot_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_1_lbnf_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_3_lbnf_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_4_lbnf_lab_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
CV_hw	127186	WGNAS	WGNAS-2024-1	Single_value	I
log_C8_NAC_4_lbnf_oth_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NEC_1_far_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NEC_3_far_sd	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C5_NAC_1_lbnf_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C5_NAC_2_lbnf_lab_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C5_NAC_2_lbnf_oth_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C5_NEC_1_far_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_2_gld_tot_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_1_lbnf_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_3_lbnf_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_4_lbnf_lab_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NAC_4_lbnf_oth_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NEC_1_far_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
log_C8_NEC_3_far_mu	127186	WGNAS	WGNAS-2024-1	Vector	I
C5_NAC_1	127186	WGNAS	WGNAS-2024-1	Matrix	C
C5_NAC_1_tot	127186	WGNAS	WGNAS-2024-1	Vector	C
C5_NAC_2	127186	WGNAS	WGNAS-2024-1	Matrix	C
C5_NAC_2_lab	127186	WGNAS	WGNAS-2024-1	Vector	C
C5_NAC_2_other	127186	WGNAS	WGNAS-2024-1	Vector	C
C5_NEC_1	127186	WGNAS	WGNAS-2024-1	Matrix	C
C5_NEC_1_tot	127186	WGNAS	WGNAS-2024-1	Vector	C

Chapter 4

WGNAS

The WGNAS database is created in schemas refnas and datna (see Section 1).
This section contains the code to import WGNAS to the DIADROMOUS DB.

4.1 Create referential for WGNAS

Creating the referential for WGNAS

```
DROP TABLE IF EXISTS refnas.tr_version_ver CASCADE;
CREATE TABLE refnas.tr_version_ver() inherits (ref.tr_version_ver);

ALTER TABLE refnas.tr_version_ver ADD CONSTRAINT ver_code_pkey PRIMARY KEY (ver_code);
ALTER TABLE refnas.tr_version_ver ADD CONSTRAINT  fk_ver_spe_code FOREIGN KEY (ver_spe_code)
REFERENCES ref.tr_species_spe(spe_code)
ON UPDATE CASCADE ON DELETE CASCADE;

COMMENT ON TABLE refnas.tr_version_ver
IS 'Table of data or variable version, essentially one datacall or advice, inherits ref.tr_v

COMMENT ON COLUMN refnas.tr_version_ver.ver_code
IS 'Version code, stockkey-year-version.';
COMMENT ON COLUMN refnas.tr_version_ver.ver_year
IS 'Year of assessement.';
COMMENT ON COLUMN refnas.tr_version_ver.ver_spe_code
IS 'Species code e.g. ''127186'' references tr_species_spe.';
COMMENT ON COLUMN refnas.tr_version_ver.ver_stockkeylabel
IS 'Ver_stockkeylabel e.g. ele.2737.nea.';
COMMENT ON COLUMN refnas.tr_version_ver.ver_datacalldoi
IS 'Data call DOI, find a way to retrieve that information
and update this comment';
```

```
COMMENT ON COLUMN refnas.tr_version_ver.ver_version
IS 'Version code in original database, eg 2,4 for wgnas, dc_2020 for wgeel.';
COMMENT ON COLUMN refnas.tr_version_ver.ver_description
IS 'Description of the data call / version.';
GRANT ALL ON refnas.tr_version_ver TO diaspara_admin;
GRANT SELECT ON refnas.tr_version_ver TO diaspara_read;
```

4.2 Import the metadata table

Creating the referential for WGNAS

```
DROP TABLE IF EXISTS datnas.t_metadata_met;

CREATE TABLE datnas.t_metadata_met(met_oldversion numeric)
INHERITS (ref.t_metadata_met);


-- ADDING CONSTRAINTS

ALTER TABLE datnas.t_metadata_met ADD
 CONSTRAINT t_metadata_met_pkey PRIMARY KEY(met_var, met_spe_code);

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_spe_code FOREIGN KEY (met_spe_code)
  REFERENCES ref.tr_species_spe(spe_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

 ALTER TABLE datnas.t_metadata_met ADD
    CONSTRAINT ck_met_spe_code CHECK (met_spe_code='127186');

 ALTER TABLE datnas.t_metadata_met ADD
    CONSTRAINT fk_met_wkg_code FOREIGN KEY (met_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;
```

```
ALTER TABLE datnas.t_metadata_met ADD
    CONSTRAINT ck_met_wkg_code CHECK (met_wkg_code='WGNAS');

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_ver_code FOREIGN KEY (met_ver_code)
  REFERENCES refnas.tr_version_ver(ver_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_oty_code FOREIGN KEY (met_oty_code)
  REFERENCES ref.tr_objecttype_oty (oty_code) ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_nim_code FOREIGN KEY (met_nim_code)
  REFERENCES ref.tr_nimble_nim (nim_code) ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_mtr_code FOREIGN KEY (met_mtr_code)
  REFERENCES ref.tr_metric_mtr(mtr_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_uni_code FOREIGN KEY (met_uni_code)
  REFERENCES ref.tr_units_uni(uni_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_cat_code FOREIGN KEY (met_cat_code)
  REFERENCES ref.tr_category_cat(cat_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;

ALTER TABLE datnas.t_metadata_met ADD
  CONSTRAINT fk_met_des_code FOREIGN KEY (met_des_code)
  REFERENCES ref.tr_destination_des(des_code)
  ON DELETE CASCADE
  ON UPDATE CASCADE;
--  COMMENTS FOR WGNAS
```

```
COMMENT ON TABLE datnas.t_metadata_met IS
'Table (metadata) of each variable (parameter) in the wgnas database.';
COMMENT ON COLUMN refnas.t_metadata_met.met_var
IS 'Variable code, primary key on both met_spe_code and met_var.';
COMMENT ON COLUMN refnas.t_metadata_met.met_spe_code
IS 'Species, 127186 primary key on both met_spe_code and met_var.';
COMMENT ON COLUMN refnas.t_metadata_met.met_ver_code
IS 'Code on the version of the model, see table tr_version_ver.';
COMMENT ON COLUMN refnas.t_metadata_met.met_oty_code
IS 'Object type, single_value, vector, matrix see table tr_objecttype_oty.';
COMMENT ON COLUMN refnas.t_metadata_met.met_nim_code
IS 'Nimble type, one of data, constant, output, other.';
COMMENT ON COLUMN refnas.t_metadata_met.met_dim
IS 'Dimension of the Nimble variable, use {10, 100, 100}
to insert the description of an array(10,100,100).';
COMMENT ON COLUMN refnas.t_metadata_met.met_dimname
IS 'Dimension of the variable in Nimble, use {''year'', ''stage'', ''area''}.';
COMMENT ON COLUMN refnas.t_metadata_met.met_modelstage
IS 'Currently one of fit, other, First year.';
COMMENT ON COLUMN refnas.t_metadata_met.met_type
IS 'Type of data in the variable, homewatercatches, InitialISation first year,
abundance ....';
COMMENT ON COLUMN refnas.t_metadata_met.met_location
IS 'Describe process at sea, e.g. Btw. FAR - GLD fisheries, or Aft. Gld fISheries.';
COMMENT ON COLUMN refnas.t_metadata_met.met_fishery
IS 'Description of the fishery.';
COMMENT ON COLUMN refnas.t_metadata_met.met_des_code
IS 'Destination of the fish, e.g. Released (alive), Seal damage,
Removed (from the environment), references table tr_destination_des., this is currently only
so can be kept NULL';
COMMENT ON COLUMN refnas.t_metadata_met.met_uni_code
IS 'Unit, refnaserences table tr_unit_uni.';
COMMENT ON COLUMN refnas.t_metadata_met.met_cat_code
IS 'Broad category of data or parameter,
catch, effort, biomass, mortality, count ...refnaserences table tr_category_cat.';
COMMENT ON COLUMN refnas.t_metadata_met.met_mtr_code
IS 'Code of the metric, refnaserences tr_metric_mtr, Estimate, Bound, SD, CV ....';
COMMENT ON COLUMN refnas.t_metadata_met.met_definition
IS 'Definition of the metric.';
COMMENT ON COLUMN refnas.t_metadata_met.met_deprecated
IS'Is the variable still used ?';


ALTER TABLE datnas.t_metadata_met OWNER TO diaspara_admin;
GRANT SELECT ON datnas.t_metadata_met TO diaspara_read;
```

```r
# t_metadata_met

metadata <- dbGetQuery(con_salmoglob, "SELECT * FROM metadata")

res <- dbGetQuery(con_diaspara, "SELECT * FROM datnas.t_metadata_met;")
#clipr::write_clip(colnames(res))

# unique(metadata$metric)
# dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_metric_mtr")

t_metadata_met <-
  data.frame(
    met_var = metadata$var_mod,
    met_spe_code = "127186",
    met_wkg_code = "WGNAS",
    met_ver_code = "SAL-2024-1", # no data on version in metadata
    met_oty_code = metadata$type_object,
    met_nim_code =  case_when(
      "Data_nimble"== metadata$nimble ~ "Data",
      "Const_nimble" == metadata$nimble ~ "Parameter constant",
      "Output" == metadata$nimble ~ "Output",
      "other" == metadata$nimble ~ "Other",
      .default = NA),
    met_dim = paste0(
      "{", metadata$dim1, ",",
      replace_na(metadata$dim2, 0), ",",
      replace_na(metadata$dim3, 0), "}"
    ),
    met_dimname = paste0(
      "{'", metadata$name_dim1, "',",
      ifelse(metadata$name_dim2 == "", "NULL", paste0("'", metadata$name_dim2, "'")), ",",
      ifelse(metadata$name_dim3 == "", "NULL", paste0("'", metadata$name_dim3, "')), "}"
    ),
    met_modelstage = metadata$model_stage,
    met_type = metadata$type,
    met_location = metadata$locations,
    met_fishery = metadata$fishery,
    met_mtr_code = case_when(metadata$metric == "Standard deviation" ~ "SD",
                             metadata$metric == "Coefficient of variation" ~ "CV",
                             .default = metadata$metric
```

```
    ),
    met_des_code = NA,
    met_uni_code = NA, # (TODO)
    met_cat_code = case_when(
      grepl("Origin distribution in sea catches", metadata$type) ~ "Other",
      grepl("catch", metadata$type) ~ "Catch",
      grepl("harvest rates", metadata$type) ~ "Mortality",
      grepl("Survival rate", metadata$type) ~ "Mortality",
      grepl("Returns", metadata$type) ~ "Count",
      grepl("Fecundity", metadata$type) ~ "Life trait",
      grepl("Sex ratio", metadata$type) ~ "Life trait",
      grepl("Maturation rate", metadata$type) ~ "Life trait",
      grepl("Proportion", metadata$type) ~ "Other",
      grepl("Stocking", metadata$type) ~ "Count",
      grepl("Smolt age structure", metadata$type) ~ "Life trait",
      grepl("Time spent", metadata$type) ~ "Life trait",
      grepl("Conservation limits", metadata$type) ~ "Conservation limit",
      grepl("Abundance", metadata$type) ~ "Count",
      grepl("Demographic transitions", metadata$type) ~ "Other",
      grepl("year", metadata$type) ~ "Other",
      grepl("Number of SU", metadata$type) ~ "Other",
      grepl("Prior", metadata$type) ~ "Other",
      grepl("Number of SU", metadata$type) ~ "Other",
      .default = NA
    ),
    met_definition = metadata$definition,
    met_deprecated = NA

  )

res <- dbWriteTable(con_diaspara_admin, "t_metadata_met_temp",
                    t_metadata_met, overwrite = TRUE)
dbExecute(con_diaspara_admin, "INSERT INTO datnas.t_metadata_met
SELECT
 met_var,
 met_spe_code,
 met_wkg_code,
 met_ver_code,
 upper(substring(met_oty_code from 1 for 1)) ||
          substring(met_oty_code from 2 for length(met_oty_code)),
 met_nim_code,
 met_dim::INTEGER[],
 met_dimname::TEXT[],
 met_modelstage,
 met_type,
```

```
 met_location,
 met_fishery,
 met_mtr_code,
 met_des_code,
 met_uni_code,
 met_cat_code,
 met_definition,
 met_deprecated
FROM t_metadata_met_temp")

dbExecute(con_diaspara_admin, "DROP TABLE t_metadata_temp CASCADE;")
```

After integration, the table of metadata from WGNAS is not changed much, apart from adapting to the new referentials. The table is shown in Table 4.1 below.

```
dbGetQuery(con_diaspara, "SELECT * FROM datnas.t_metadata_met limit 10;")|> knitr::kable()
```

met_var	met_spe_code	met_wkg_code	met_ver_code	met_oty_code	m
log_C5_NAC_1_lbnf_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C5_NAC_2_lbnf_lab_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C5_NAC_2_lbnf_oth_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C5_NEC_1_far_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C8_2_gld_tot_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C8_NAC_1_lbnf_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C8_NAC_3_lbnf_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
log_C8_NAC_4_lbnf_lab_sd	127186	WGNAS	WGNAS-2024-1	Vector	P
CV_hw	127186	WGNAS	WGNAS-2024-1	Single_value	P
log_C8_NAC_4_lbnf_oth_sd	127186	WGNAS	WGNAS-2024-1	Vector	P

4.3 Import the `database` table from the salmoglob (WGNAS) database

For data we first need to create the table.

The code for creating `t_stock_sto` is listed below, this table is created for all working groups, it should not have any lines, only get those from inheritance from schema datnas, datang, datbast, …

SQL code to create tables

```
-- before working there should have been these constraints in the salmoglob DB
```

```
 ALTER TABLE "database"
ADD CONSTRAINT c_uk_area_varmod_year_location_age UNIQUE  (area, var_mod, "year", "location"
ALTER TABLE "database_archive" ADD CONSTRAINT c_uk_archive_version_area_varmod_year_locatio
UNIQUE  ("version", area, var_mod, "year", "location", age);

-- For the archive db, the constraint is not working meaning that we have duplicated values

SELECT DISTINCT met_nim_code FROM datnas.t_metadata_met
JOIN refsalmoglob."database" ON var_mod = met_var
WHERE  met_cat_code ='Other'


SELECT * FROM datnas.t_metadata_met WHERE met_var LIKE '%mu%'
SELECT DISTINCT met_modelstage FROM datnas.t_metadata_met

-- This will create the main table to hold the stock data
-- I'm currenlty putting foreign key to ref but this is just for show because this table
-- will only contain inherited valeus


DROP TABLE IF EXISTS dat.t_stock_sto CASCADE;
CREATE TABLE dat.t_stock_sto (
  sto_id SERIAL NOT NULL,
  sto_met_var TEXT NOT NULL,
  sto_year INT4 NULL,
  sto_spe_code VARCHAR(3) NOT NULL,
  CONSTRAINT fk_sto_met_var_met_spe_code
    FOREIGN KEY (sto_met_var, sto_spe_code) REFERENCES dat.t_metadata_met(met_var,met_spe_co
    ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_value NUMERIC NULL,
  sto_are_code TEXT NOT NULL,
  CONSTRAINT fk_sto_are_code FOREIGN KEY (sto_are_code)
    REFERENCES "ref".tr_area_are (are_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  -- NOTE : here I'm referencing the code because it's more easy to grasp than a number, but
  -- should work stil but requires a unique constraint on code (which we have set).
  sto_cou_code VARCHAR(2) NULL,
  CONSTRAINT fk_sto_cou_code FOREIGN KEY (sto_cou_code)
    REFERENCES "ref".tr_country_cou (cou_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_lfs_code TEXT NOT NULL,
  CONSTRAINT fk_sto_lfs_code_sto_spe_code FOREIGN KEY (sto_lfs_code, sto_spe_code)
    REFERENCES "ref".tr_lifestage_lfs (lfs_code, lfs_spe_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_hty_code VARCHAR(2) NULL,
```

```
  CONSTRAINT fk_hty_code FOREIGN KEY (sto_hty_code)
    REFERENCES "ref".tr_habitattype_hty(hty_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  --sto_fia_code TEXT NULL,
  --CONSTRAINT fk_sto_fia_code FOREIGN KEY(sto_fia_code)
  --  REFERENCES "ref".tr_fishingarea_fia(fia_code)
  --  ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_qal_code INT4 NOT NULL,
  CONSTRAINT fk_sto_qal_code FOREIGN KEY (sto_qal_code)
    REFERENCES "ref".tr_quality_qal(qal_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_qal_comment TEXT NULL,
  sto_comment TEXT NULL,
  sto_datelastupdate date NULL,
  sto_mis_code VARCHAR(2) NULL,
  CONSTRAINT fk_sto_mis_code FOREIGN KEY (sto_mis_code)
  REFERENCES "ref".tr_missvalueqal_mis (mis_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_dta_code TEXT DEFAULT 'Public' NULL,
  CONSTRAINT fk_dta_code FOREIGN KEY (sto_dta_code)
  REFERENCES "ref".tr_dataaccess_dta(dta_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_wkg_code TEXT NOT NULL,
  CONSTRAINT fk_sto_wkg_code  FOREIGN KEY (sto_wkg_code)
  REFERENCES "ref".tr_icworkinggroup_wkg(wkg_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT uk_sto_id_sto_wkg_code UNIQUE (sto_id, sto_wkg_code),
  CONSTRAINT ck_notnull_value_and_mis_code CHECK ((((sto_mis_code IS NULL) AND (sto_value IS
  ((sto_mis_code IS NOT NULL) AND (sto_value IS NULL)))),
  sto_ver_code TEXT ,
  CONSTRAINT fk_sto_ver_code FOREIGN KEY (sto_ver_code)
  REFERENCES ref.tr_version_ver (ver_code)
  ON UPDATE CASCADE ON DELETE RESTRICT
  -- We removed qual_id = 0
  -- CONSTRAINT ck_qal_id_and_missvalue CHECK (((eel_missvaluequal IS NULL) OR (eel_qal_id <
  -- TODO CHECK LATER HOW TO DEAL WITH DEPRECATED
  -- CONSTRAINT ck_removed_typid CHECK (((COALESCE(eel_qal_id, 1) > 5) OR (eel_typ_id <> ALL
);



COMMENT ON TABLE dat.t_stock_sto IS
'Table including the stock data from the different schema, dateel, datnas.... This table sho
 it''s getting its content by inheritance from other tables in other schema, will probably b
 by a view in SQL server';
```

```
COMMENT ON COLUMN dat.t_stock_sto.sto_id IS 'Integer serial identifying. Only unique in this
when looking at the pair, sto_id, sto_wkg_code';
COMMENT ON COLUMN dat.t_stock_sto.sto_met_var IS 'Name of the variable in the database, this
var_mod in the salmoglob database and eel_typ_id in the wgeel database, there is a unicity c
on the pair of column sto_spe_code, sto_met_code';
-- note if we end up with a single table, then the constraint will  have to be set
-- on sto_wkg_code, sto_spe_code and sto_met_code.
COMMENT ON COLUMN dat.t_stock_sto.sto_year IS 'Year';
COMMENT ON COLUMN dat.t_stock_sto.sto_value IS 'Value if null then provide a value in sto_mi
COMMENT ON COLUMN dat.t_stock_sto.sto_are_code IS 'Code of the area, areas are geographical
see tr_area_are.';
COMMENT ON COLUMN dat.t_stock_sto.sto_cou_code IS 'Code of the country see tr_country_cou, n
COMMENT ON COLUMN dat.t_stock_sto.sto_lfs_code IS 'Code of the lifestage see tr_lifestage_lf
both lfs_code, and lfs_spe_code (as two species can have the same lifestage code.';
COMMENT ON COLUMN dat.t_stock_sto.sto_hty_code IS 'Code of the habitat type, one of MO (mari
T (Transitional water), FW (Freshwater), null accepted';
-- COMMENT ON COLUMN dat.t_stock_sto.sto_fia_code IS 'For marine area, code of the ICES area
COMMENT ON COLUMN dat.t_stock_sto.sto_qal_code IS 'Code of data quality (1 good quality, 2 n
3 bad quality (not used), 4 dubious, 18, 19 ... historical data not used.
Not null, Foreign key set to tr_quality_qal';
COMMENT ON COLUMN dat.t_stock_sto.sto_qal_comment IS 'Comment for the quality, for instance
COMMENT ON COLUMN dat.t_stock_sto.sto_comment IS 'Comment on the value';
COMMENT ON COLUMN dat.t_stock_sto.sto_datelastupdate IS 'Last update of the data';
COMMENT ON COLUMN dat.t_stock_sto.sto_mis_code IS 'When no value are given in sto_value, jus
references table tr_missvalueqal_mis, should be null if value is provided (can''t have both)
COMMENT ON COLUMN dat.t_stock_sto.sto_dta_code IS 'Access to data, default is ''Public''';
COMMENT ON COLUMN dat.t_stock_sto.sto_wkg_code IS 'Code of the working group, one of
WGBAST, WGEEL, WGNAS, WKTRUTTA';
COMMENT ON COLUMN dat.t_stock_sto.sto_spe_code IS 'Code of the species';
COMMENT ON COLUMN dat.t_stock_sto.sto_ver_code IS 'Code of the version';

ALTER TABLE dat.t_stock_sto OWNER TO diaspara_admin;
GRANT SELECT ON dat.t_stock_sto  TO diaspara_read;


-- trigger on date
DROP FUNCTION dat.update_sto_datelastupdate;
CREATE OR REPLACE FUNCTION dat.update_sto_datelastupdate()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN
    NEW.sto_datelastupdate = now()::date;
    RETURN NEW;
END;
```

```
$function$
;
ALTER FUNCTION dat.update_sto_datelastupdate() OWNER TO diaspara_admin;

CREATE TRIGGER update_sto_datelastupdate BEFORE
INSERT
    OR
UPDATE
    ON
    dat.t_stock_sto FOR EACH ROW EXECUTE FUNCTION dat.update_sto_datelastupdate();

/*
 *
 * TODO CHECK THOSE TRIGGERS FOR WGEEL
 */

  /*
CREATE TRIGGER trg_check_no_ices_area AFTER
INSERT
    OR
UPDATE
    ON
    datawg.t_eelstock_eel FOR EACH ROW EXECUTE FUNCTION datawg.check_no_ices_area();
CREATE TRIGGER trg_check_the_stage AFTER
INSERT
    OR
UPDATE
    ON
    datawg.t_eelstock_eel FOR EACH ROW EXECUTE FUNCTION datawg.check_the_stage();

CREATE TRIGGER trg_check_emu_whole_aquaculture AFTER
INSERT
    OR
UPDATE
    ON
    datawg.t_eelstock_eel FOR EACH ROW EXECUTE FUNCTION datawg.checkemu_whole_country();
  */


/*
 * Added afterwards for eel
 */

/*
ALTER TABLE dat.t_stock_sto ADD COLUMN   sto_ver_code TEXT ;
```

```
ALTER TABLE dat.t_stock_sto ADD CONSTRAINT fk_sto_ver_code FOREIGN KEY (sto_ver_code)
  REFERENCES ref.tr_version_ver (ver_code)
  ON UPDATE CASCADE ON DELETE RESTRICT;
ALTER TABLE dat.t_stock_sto DROP COLUMN   sto_fia_code CASCADE;
*/
```

The same table `t_stock_sto` is created in `datnas`. It is inherited, so this means that all the column are coming from `dat.t_stock_sto` but we have to recreate all the constraints, as constraints are never inherited. Two additional check constraint are created, the value for species will always be `127186` and the value for wkg (expert group) will always be `WGNAS`.

SQL code to create tables

```
-- CREATE A TABLE INHERITED FROM dat.t_stock_sto.
-- Table dat.stock_sto only gets data by inheritance.
-- Here we have to build the constraints again.

DROP TABLE IF EXISTS datnas.t_stock_sto;
CREATE TABLE datnas.t_stock_sto (
   sto_add_code TEXT NULL,
   CONSTRAINT fk_sto_add_code FOREIGN KEY (sto_add_code)
   REFERENCES refnas.tg_additional_add (add_code),
  CONSTRAINT fk_sto_met_var_met_spe_code
    FOREIGN KEY (sto_met_var, sto_spe_code)
    REFERENCES datnas.t_metadata_met(met_var,met_spe_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_are_code FOREIGN KEY (sto_are_code)
    REFERENCES refnas.tr_area_are (are_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_cou_code FOREIGN KEY (sto_cou_code)
    REFERENCES ref.tr_country_cou (cou_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_lfs_code_sto_spe_code FOREIGN KEY (sto_lfs_code, sto_spe_code)
    REFERENCES ref.tr_lifestage_lfs (lfs_code, lfs_spe_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_hty_code FOREIGN KEY (sto_hty_code)
    REFERENCES ref.tr_habitattype_hty(hty_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
--  CONSTRAINT fk_sto_fia_code FOREIGN KEY(sto_fia_code)
--    REFERENCES ref.tr_fishingarea_fia(fia_code)
--    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_qal_code FOREIGN KEY (sto_qal_code)
    REFERENCES ref.tr_quality_qal(qal_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_mis_code FOREIGN KEY (sto_mis_code)
```

```
  REFERENCES ref.tr_missvalueqal_mis (mis_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_dta_code FOREIGN KEY (sto_dta_code)
  REFERENCES ref.tr_dataaccess_dta(dta_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_wkg_code  FOREIGN KEY (sto_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT c_uk_sto_id_sto_wkg_code UNIQUE (sto_id, sto_wkg_code),
  CONSTRAINT ck_notnull_value_and_mis_code
  CHECK ((((sto_mis_code IS NULL) AND (sto_value IS NOT NULL)) OR
  ((sto_mis_code IS NOT NULL) AND (sto_value IS NULL))))
)
inherits (dat.t_stock_sto) ;

-- This table will always be for SALMON and WGNAS

ALTER TABLE datnas.t_stock_sto ALTER COLUMN sto_spe_code SET DEFAULT '127186';
ALTER TABLE datnas.t_stock_sto ADD CONSTRAINT ck_spe_code CHECK (sto_spe_code='127186');
ALTER TABLE datnas.t_stock_sto ALTER COLUMN sto_wkg_code SET DEFAULT 'WGNAS';
ALTER TABLE datnas.t_stock_sto ADD CONSTRAINT ck_wkg_code CHECK (sto_wkg_code='WGNAS');



ALTER TABLE datnas.t_stock_sto OWNER TO diaspara_admin;
GRANT ALL ON TABLE datnas.t_stock_sto TO diaspara_read;







COMMENT ON TABLE datnas.t_stock_sto IS
'Table including the stock data in schema datnas.... This table feeds the dat.t_stock_sto ta
to the database table in the original WGNAS database.';
COMMENT ON COLUMN datnas.t_stock_sto.sto_id IS 'Integer serial identifying. Only unique in t
when looking at the pair, sto_id, sto_wkg_code';
COMMENT ON COLUMN datnas.t_stock_sto.sto_met_var IS 'Name of the variable in the database, t
var_mod in the salmoglob database and eel_typ_id in the wgeel database, there is a unicity c
on the pair of column sto_spe_code, sto_met_var';
-- note if we end up with a single table, then the constraint will  have to be set
-- on sto_wkg_code, sto_spe_code and sto_met_code.
COMMENT ON COLUMN datnas.t_stock_sto.sto_spe_code IS 'Species default ''127186'' Salmo salar
COMMENT ON COLUMN datnas.t_stock_sto.sto_year IS 'Year';
```

```
COMMENT ON COLUMN datnas.t_stock_sto.sto_value IS 'Value if null then provide a value in sto
COMMENT ON COLUMN datnas.t_stock_sto.sto_are_code IS 'Code of the area, areas are geographic
see tr_area_are.';
COMMENT ON COLUMN datnas.t_stock_sto.sto_cou_code IS 'Code of the country see tr_country_cou
COMMENT ON COLUMN datnas.t_stock_sto.sto_lfs_code IS 'Code of the lifestage see tr_lifestage
both lfs_code, and lfs_spe_code (as two species can have the same lifestage code.';
COMMENT ON COLUMN datnas.t_stock_sto.sto_hty_code IS 'Code of the habitat type, one of MO (m
T (Transitional water), FW (Freshwater), null accepted';
--COMMENT ON COLUMN datnas.t_stock_sto.sto_fia_code IS 'For marine area, code of the ICES ar
COMMENT ON COLUMN datnas.t_stock_sto.sto_qal_code IS 'Code of data quality (1 good quality,
3 bad quality (not used), 4 dubious, 18, 19 ... historical data not used.
Not null, Foreign key set to tr_quality_qal';
COMMENT ON COLUMN datnas.t_stock_sto.sto_qal_comment IS 'Comment for the quality, for instan
COMMENT ON COLUMN datnas.t_stock_sto.sto_comment IS 'Comment on the value';
COMMENT ON COLUMN datnas.t_stock_sto.sto_datelastupdate IS 'Last update of the data';
COMMENT ON COLUMN datnas.t_stock_sto.sto_mis_code IS 'When no value are given in sto_value,
references table tr_missvalueqal_mis, should be null if value is provided (can''t have both)
COMMENT ON COLUMN datnas.t_stock_sto.sto_dta_code IS 'Access to data, default is ''Public''';
COMMENT ON COLUMN datnas.t_stock_sto.sto_wkg_code IS 'Code of the working group, one of
WGBAST, WGEEL, WGNAS, WKTRUTTA';
COMMENT ON COLUMN datnas.t_stock_sto.sto_add_code IS 'Additional code in the extra dimension
collated in table tg_additional_add';


-- trigger on date
DROP FUNCTION IF EXISTS datnas.update_sto_datelastupdate;
CREATE OR REPLACE FUNCTION datnas.update_sto_datelastupdate()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN
    NEW.sto_datelastupdate = now()::date;
    RETURN NEW;
END;
$function$
;
ALTER FUNCTION datnas.update_sto_datelastupdate() OWNER TO diaspara_admin;

CREATE TRIGGER update_sto_datelastupdate BEFORE
INSERT
    OR
UPDATE
    ON
    datnas.t_stock_sto FOR EACH ROW EXECUTE FUNCTION datnas.update_sto_datelastupdate();
```

```
-- fix after change in wgeel
/*
SELECT * FROM refnas.tr_version_ver
UPDATE datnas.t_stock_sto SET sto_ver_code = 'WGNAS-2024-1'; -- 45076
*/
```

duplicated values in archive

see github duplicated values in archive DB

Some of the variables in salmoglob have no `year` dimension, this leads to dropping the non null constraint on year. We need to check for possible impact in the eel db see issue #14 : NULL values allowed for year

4.4 Table of grouping for area and age : datnas.tg_additional_add

The year column does not always contain year. In fact the database that we have created is not suited to store transfer matrix where the dimensions have area x area. We only have one area column.

This will be for parameter `omega` see location paragraph in WGNAS database description.

Another problem is the age column. When looking at the analysis see age in WGNAS database description

Only the variables `eggs`, `p_smolt`, `p_smolt_pr` and `prop_female` need an age.

This was solved using the tg_additional_add column. The stucture for WGBAST sent by Becky indicates that they too could use this additional column to store some of the matrix output.

SQL code to create tables

```
DROP TABLE IF EXISTS refnas.tg_additional_add;
CREATE TABLE refnas.tg_additional_add AS
SELECT are_code AS add_code, 'Area' AS add_type FROM refnas.tr_area_are
UNION
SELECT age_code AS add_code, 'Age' AS add_type FROM "ref".tr_age_age; --80
ALTER TABLE refnas.tg_additional_add ADD CONSTRAINT
uk_add_code UNIQUE (add_code);
```

```
ALTER TABLE refnas.tg_additional_add OWNER TO diaspara_admin;
GRANT ALL ON TABLE refnas.tg_additional_add TO diaspara_read;




COMMENT ON TABLE refnas.tg_additional_add IS
'Table including the stock data in schema datnas.... This table feeds the dat.t_stock_sto ta
to the database table in the original WGNAS database.';
COMMENT ON COLUMN refnas.tg_additional_add.add_code IS 'Code coming from are_code in
table refnas.tr_area_are or age_code in table ref.tr_age_age';
COMMENT ON COLUMN refnas.tg_additional_add.add_type IS 'One of Area or Age';
```

Table 4.2: Content of the refnas additional table

add_code	add_type
RU_KB	Area
Atlantic	Area
GF	Area
Denmark	Area
coun_Labrador	Area
RU_RP	Area
2FW	Age
coun_France	Area
coun_Finland	Area
NEAC	Area
NO_SW	Area
Finland	Area
NEAC inland	Area
MSW	Age
coun_Scotland	Area
NI_FO	Area
LB fishery	Area
coun_Gulf	Area
6FW	Age
Iceland	Area
NF	Area
coun_US	Area
Netherlands	Area
IC_SW	Area
NI_FB	Area
coun_Iceland_NE	Area

NEC	Area
coun_Iceland_SW	Area
Svalbard and Jan Mayen	Area
Great Britain	Area
coun_Russia	Area
QC	Area
3FW	Age
NO_NO	Area
IR	Area
1FW	Age
5FW	Age
coun_Ireland	Area
2SW	Age
Germany	Area
Russia	Area
Sweden	Area
RU_AK	Area
neNF fishery	Area
FR	Area
FAR fishery	Area
coun_Scotia Fundy	Area
FI	Area
4FW	Age
GLD fishery	Area
SC_EA	Area
coun_England_Wales	Area
Portugal	Area
NO_MI	Area
coun_Sweden	Area
Ireland	Area
coun_Norway	Area
Luxembourg	Area
LB	Area
NAC	Area
Belgium	Area
coun_Newfoundland	Area
LB/SPM/swNF fishery	Area
SC_WE	Area
coun_Northern_Ireland	Area
coun_Quebec	Area
France	Area
NEAC marine	Area

1SW	Age
SW	Area
0FW	Age
EW	Area
US	Area
Spain	Area
NO_SE	Area
Czech republic	Area
RU_KW	Area
SF	Area
Norway	Area
IC_NE	Area

4.5 Import the t_stock_sto

```
dbExecute(con_diaspara,"ALTER SEQUENCE dat.t_stock_sto_sto_id_seq RESTART WITH 1;")
dbExecute(con_diaspara_admin,"DELETE FROM datnas.t_stock_sto;")
dbExecute(con_diaspara_admin,"INSERT INTO datnas.t_stock_sto
(sto_id, sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code,
sto_cou_code, sto_lfs_code, sto_hty_code, sto_qal_code,
sto_qal_comment, sto_comment, sto_datelastupdate, sto_mis_code,
sto_dta_code, sto_wkg_code,sto_add_code)
SELECT
nextval('dat.t_stock_sto_sto_id_seq'::regclass) AS sto_id
, d.var_mod AS sto_met_var
, d.year AS sto_year
, '127186' AS  sto_spe_code
, d.value AS sto_value
, d.area AS sto_are_code
, NULL AS sto_cou_code -- OK can be NULL
, CASE WHEN m.life_stage = 'Eggs' THEN 'E'
    WHEN m.life_stage = 'Adult' THEN 'A'
    WHEN m.life_stage = 'Multiple' THEN 'AL'
    WHEN m.life_stage = 'Adults' THEN 'A'
    WHEN m.life_stage = 'Smolts' THEN 'SM'
    WHEN m.life_stage = 'Non mature' THEN 'PS' -- IS THAT RIGHT ?
    WHEN m.life_stage = 'PFA' THEN 'PS' -- No VALUES
    WHEN m.life_stage = 'Spawners' THEN 'A' -- No values
    WHEN m.life_stage = '_' THEN '_'
   ELSE 'TROUBLE' END AS sto_lfs_code
, NULL AS sto_hty_code
, 1 AS sto_qal_code -- see later TO INSERT deprecated values
, NULL AS sto_qal_comment
```

```
, NULL AS sto_comment
, date(d.date_time) AS sto_datelastupdate
, NULL AS sto_mis_code
, 'Public' AS sto_dta_code
, 'WGNAS' AS sto_wkg_code
, CASE WHEN d.var_mod IN ('eggs','p_smolt', 'p_smolt_pr', 'prop_female') THEN d.age
       WHEN d.var_mod IN ('omega') THEN d.LOCATION
       END AS sto_add_code
FROM refsalmoglob.database d JOIN
refsalmoglob.metadata m ON m.var_mod = d.var_mod; ")# 45076
```

4.6 structure of the table datnas.t_stock_sto

```
dbGetQuery(con_diaspara, "SELECT * FROM datnas.t_stock_sto limit 10;")|>
  knitr::kable() |>
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

sto_id	sto_met_var	sto_year	sto_spe_code	sto_value	sto_are_code	sto_cou_code	sto_lfs_
21254	E_theta1	2014	127186	0.007	QC	NA	E
21255	E_theta1	2013	127186	0.007	QC	NA	E
21256	E_theta1	2012	127186	0.007	QC	NA	E
21257	E_theta1	2011	127186	0.007	QC	NA	E
21258	E_theta1	2010	127186	0.007	QC	NA	E
21259	E_theta1	2009	127186	0.007	QC	NA	E
21260	E_theta1	2008	127186	0.007	QC	NA	E
21261	E_theta1	2007	127186	0.007	QC	NA	E
21262	E_theta1	2006	127186	0.007	QC	NA	E
21263	E_theta1	2005	127186	0.007	QC	NA	E

Chapter 5

WGEEL

The main difficulty in transfering the WGEEL database to the DIADROMOUS database lies in the way the data are linked to areas in the marine habitat. Currently we have kept all historical data with the "historical" EMU as the reference. This will probably change once we start to build the model and the habitat DB proposes a structure of eel habitat

5.1 refeel.tr_version_ver

See metricDB report

5.2 dateel.t_metadata_met

SQL code to create table `dateel.t_metadata_met`

```
DROP TABLE IF EXISTS dateel.t_metadata_met;

CREATE TABLE dateel.t_metadata_met(
 CONSTRAINT t_metadata_met_pkey PRIMARY KEY(met_var, met_wkg_code),
 CONSTRAINT fk_met_spe_code FOREIGN KEY (met_spe_code)
  REFERENCES ref.tr_species_spe(spe_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
 CONSTRAINT ck_met_spe_code CHECK (met_spe_code='126281'),
 CONSTRAINT fk_met_wkg_code FOREIGN KEY (met_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT ck_met_wkg_code CHECK (met_wkg_code='WGEEL'),
  CONSTRAINT fk_met_ver_code FOREIGN KEY (met_ver_code)
  REFERENCES refeel.tr_version_ver(ver_code)
```

```
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_oty_code FOREIGN KEY (met_oty_code)
  REFERENCES ref.tr_objecttype_oty (oty_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_nim_code FOREIGN KEY (met_nim_code)
  REFERENCES ref.tr_nimble_nim (nim_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_mtr_code FOREIGN KEY (met_mtr_code)
  REFERENCES ref.tr_metric_mtr(mtr_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_uni_code FOREIGN KEY (met_uni_code)
  REFERENCES ref.tr_units_uni(uni_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_cat_code FOREIGN KEY (met_cat_code)
  REFERENCES ref.tr_category_cat(cat_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_des_code FOREIGN KEY (met_des_code)
  REFERENCES ref.tr_destination_des(des_code)
  ON UPDATE CASCADE ON DELETE RESTRICT
)
INHERITS (dat.t_metadata_met);



--  COMMENTS FOR WGEEL

COMMENT ON TABLE dateel.t_metadata_met IS
'Table (metadata) of each variable (parameter) in the wgeel database.';
COMMENT ON COLUMN dateel.t_metadata_met.met_var
IS 'Variable code, primary key on both met_spe_code and met_var.';
COMMENT ON COLUMN dateel.t_metadata_met.met_spe_code
IS 'Species, ''126281'' primary key on both met_spe_code and met_var.';
COMMENT ON COLUMN dateel.t_metadata_met.met_ver_code
IS 'Code on the version of the model, see table refeel.tr_version_ver.';
COMMENT ON COLUMN dateel.t_metadata_met.met_oty_code
IS 'Object type, single_value, vector, matrix see table tr_objecttype_oty.';
COMMENT ON COLUMN dateel.t_metadata_met.met_nim_code
IS 'Nimble type, one of data, constant, output, other.';
COMMENT ON COLUMN dateel.t_metadata_met.met_dim
IS 'Dimension of the Nimble variable, use {10, 100, 100}
to insert the description of an array(10,100,100).';
COMMENT ON COLUMN dateel.t_metadata_met.met_dimname
IS 'Dimension of the variable in Nimble, use {''year'', ''stage'', ''area''}.';
COMMENT ON COLUMN dateel.t_metadata_met.met_modelstage
IS 'Currently one of fit, other, First year.';
```

```
COMMENT ON COLUMN dateel.t_metadata_met.met_type
IS 'Type of data in the variable, homewatercatches, InitialISation first year,
abundance ....';
COMMENT ON COLUMN dateel.t_metadata_met.met_location
IS 'Describe process with geographical information';
COMMENT ON COLUMN dateel.t_metadata_met.met_fishery
IS 'Description of the fishery.';
COMMENT ON COLUMN dateel.t_metadata_met.met_des_code
IS 'Destination of the fish, e.g. Released (alive), Seal damage,
Removed (from the environment), references table tr_destination_des., this is currently only
so can be kept NULL';
COMMENT ON COLUMN dateel.t_metadata_met.met_uni_code
IS 'Unit, dateelerences table tr_unit_uni.';
COMMENT ON COLUMN dateel.t_metadata_met.met_cat_code
IS 'Broad category of data or parameter,
catch, effort, biomass, mortality, count ...dateelerences table tr_category_cat.';
COMMENT ON COLUMN dateel.t_metadata_met.met_mtr_code
IS 'Code of the metric, dateelerences tr_metric_mtr, Estimate, Bound, SD, CV ....';
COMMENT ON COLUMN dateel.t_metadata_met.met_definition
IS 'Definition of the metric.';
COMMENT ON COLUMN dateel.t_metadata_met.met_deprecated
IS'Is the variable still used ?';



ALTER TABLE dateel.t_metadata_met OWNER TO diaspara_admin;
GRANT SELECT ON dateel.t_metadata_met TO diaspara_read;
```

We currently consider that SumH, and Biom are “Output”, the result of model.

type is not a referential, but used for legacy in WGNAS see type table so it’s currently empty in the table

```
# t_metadata_met

eelstock <- dbGetQuery(con_diaspara_admin, "SELECT * FROM datwgeel.t_eelstock_eel WHERE eel_
nrow(eelstock) # 73730
unique(eelstock$eel_typ_id)
# 6  4  8  9 11 17 18 15 14 13 16 19 10 32 33 34
```

```r
# View(eelstock[eelstock$eel_typ_id ==32,])



res <- dbGetQuery(con_diaspara, "SELECT * FROM dateel.t_metadata_met;")
clipr::write_clip(colnames(res))



typ <- dbGetQuery(con_diaspara_admin,"SELECT *  FROM refwgeel.tr_typeseries_typ")
# below I'm removing from typ as these values are not actually in the database
typ <- typ[!typ$typ_id %in% c(1,2,3),]  # remove series
typ <- typ[!typ$typ_id %in% c(16),]  # potential_availabe_habitat_production_ha
typ <- typ[!typ$typ_id %in% c(5, 7),]  # com_catch and rec_catch
typ <- typ[!typ$typ_id %in% c(26:31),]  # silver eel equivalents (deprecated)
# unique(metadata$metric)
# dbGetQuery(con_diaspara, "SELECT * FROM ref.tr_metric_mtr")
View(typ)
t_metadata_met <-
  data.frame(
    met_var = typ$typ_name,
    met_spe_code = "126281",
    met_wkg_code = "WGEEL",
    met_ver_code = "WGEEL-2025-1",
    met_oty_code = "Single_value", # https://diaspara.bordeaux-aquitaine.inrae.fr/deliverabl
    met_nim_code =  case_when(
      typ$typ_id %in% c(4:12,32,33)   ~ "Data",
      .default = "Output"), # https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/
    met_dim = paste0(
      "{", 1, ",",
       0, ",",
       0, "}"
    ),
    met_dimname = paste0(
      "{'year',NULL,NULL}"
    ),
    met_modelstage = NA,
    met_type = typ$typ_id,
    # not a referential, used for legacy in WGNAS, and I'm using the old code in wgeel
    # see https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p4/wgnas_salmoglob_c
    met_location = NA, # something line bef. Fisheries Aft fisheries.... not a referential
    met_fishery = NA, # not a referential
    met_mtr_code = NA, # reference to tr_metrictype (bound, mean, SD, can be left empty)
    met_des_code = NA,
    met_uni_code = NA, # (TODO)
```

```
    met_cat_code = case_when(
typ$typ_name == "com_landings_kg" ~ "Catch",
typ$typ_name == "rec_landings_kg" ~ "Catch",
typ$typ_name == "other_landings_kg" ~ "Catch",
typ$typ_name == "other_landings_n" ~ "Catch",
typ$typ_name == "gee_n" ~ "Count",
typ$typ_name == "q_aqua_kg" ~ "Other" ,
typ$typ_name == "q_aqua_n" ~ "Other" ,
typ$typ_name == "q_release_kg" ~ "Release",
typ$typ_name == "q_release_n" ~ "Release",
typ$typ_name == "b0_kg" ~ "Biomass",
typ$typ_name == "bbest_kg" ~ "Biomass",
typ$typ_name == "b_current_without_stocking_kg" ~ "Biomass",
typ$typ_name == "bcurrent_kg" ~ "Biomass",
typ$typ_name == "suma" ~ "Mortality",
typ$typ_name == "sumf" ~ "Mortality",
typ$typ_name == "sumh" ~ "Mortality",
typ$typ_name == "sumf_com" ~ "Mortality",
typ$typ_name == "sumf_rec" ~ "Mortality",
typ$typ_name == "sumh_hydro" ~ "Mortality",
typ$typ_name == "sumh_habitat" ~ "Mortality",
typ$typ_name == "sumh_other" ~ "Mortality",
typ$typ_name == "sumh_release" ~ "Mortality",
.default = NA
    ),
met_definition = typ$typ_description,
met_deprecated = NA
# not integrating any of the deprecated data
)

res <- dbWriteTable(con_diaspara_admin, "t_metadata_met_wgeel_temp",
                    t_metadata_met, overwrite = TRUE)
dbExecute(con_diaspara_admin, "INSERT INTO dateel.t_metadata_met
SELECT
 met_var,
 met_spe_code,
 met_wkg_code,
 met_ver_code,
 met_oty_code,
 met_nim_code,
 met_dim::INTEGER[],
 met_dimname::TEXT[],
 met_modelstage,
 met_type,
 met_location,
```

```
 met_fishery,
 met_mtr_code,
 met_des_code,
 met_uni_code,
 met_cat_code,
 met_definition,
 met_deprecated
FROM t_metadata_met_wgeel_temp") # 22

dbExecute(con_diaspara_admin, "DROP TABLE t_metadata_met_wgeel_temp CASCADE;")
```

> **TODO DIASPARA**
>
> We still need to add units to the metadata table

5.3 dateel.t_stock_sto

SQL code to create table `dateel.t_stock_sto`

```
-- CREATE A TABLE INHERITED FROM dat.t_stock_sto.
-- Table dat.stock_sto only gets data by inheritance.
-- Here we have to build the constraints again.

DROP TABLE IF EXISTS dateel.t_stock_sto;
CREATE TABLE dateel.t_stock_sto (
  CONSTRAINT fk_sto_met_var_met_spe_code
    FOREIGN KEY (sto_met_var, sto_spe_code) REFERENCES dateel.t_metadata_met(met_var,met_spe
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_are_code FOREIGN KEY (sto_are_code)
    REFERENCES refeel.tr_area_are (are_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_cou_code FOREIGN KEY (sto_cou_code)
    REFERENCES ref.tr_country_cou (cou_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_lfs_code_sto_spe_code FOREIGN KEY (sto_lfs_code, sto_spe_code)
    REFERENCES ref.tr_lifestage_lfs (lfs_code, lfs_spe_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_hty_code FOREIGN KEY (sto_hty_code)
    REFERENCES ref.tr_habitattype_hty(hty_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  --CONSTRAINT fk_sto_fia_code FOREIGN KEY(sto_fia_code)
  --  REFERENCES ref.tr_fishingarea_fia(fia_code)
  --  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_qal_code FOREIGN KEY (sto_qal_code)
```

```
    REFERENCES ref.tr_quality_qal(qal_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_mis_code FOREIGN KEY (sto_mis_code)
  REFERENCES ref.tr_missvalueqal_mis (mis_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_dta_code FOREIGN KEY (sto_dta_code)
  REFERENCES ref.tr_dataaccess_dta(dta_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_wkg_code  FOREIGN KEY (sto_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT c_uk_sto_id_sto_wkg_code UNIQUE (sto_id, sto_wkg_code),
  CONSTRAINT ck_notnull_value_and_mis_code CHECK ((((sto_mis_code IS NULL) AND (sto_value IS
  ((sto_mis_code IS NOT NULL) AND (sto_value IS NULL))))
)
inherits (dat.t_stock_sto) ;

-- This table will always be for EEL (ang) and WGEEL

ALTER TABLE dateel.t_stock_sto ALTER COLUMN sto_spe_code SET DEFAULT '126281';
ALTER TABLE dateel.t_stock_sto ADD CONSTRAINT ck_spe_code CHECK (sto_spe_code='126281');
ALTER TABLE dateel.t_stock_sto ALTER COLUMN sto_wkg_code SET DEFAULT 'WGEEL';
ALTER TABLE dateel.t_stock_sto ADD CONSTRAINT ck_wkg_code CHECK (sto_wkg_code='WGEEL');



ALTER TABLE dateel.t_stock_sto OWNER TO diaspara_admin;
GRANT ALL ON TABLE dateel.t_stock_sto TO diaspara_read;









COMMENT ON TABLE dateel.t_stock_sto IS
'Table including the stock data in schema dateel.... This table feeds the dat.t_stock_sto ta
to the t_eelstock_eel table in the original WGEEL database.';
COMMENT ON COLUMN dateel.t_stock_sto.sto_id IS 'Integer serial identifying. Only unique in t
when looking at the pair, sto_id, sto_wkg_code';
COMMENT ON COLUMN dateel.t_stock_sto.sto_met_var IS 'Name of the variable in the database, t
eel_typ_name in the eel database, there is a unicity constraint based
on the pair of column sto_spe_code, sto_met_var';
-- note if we end up with a single table, then the constraint will  have to be set
-- on sto_wkg_code, sto_spe_code and sto_met_code.
```

```
COMMENT ON COLUMN dateel.t_stock_sto.sto_year IS 'Year';
COMMENT ON COLUMN dateel.t_stock_sto.sto_value IS 'Value if null then provide a value in sto
COMMENT ON COLUMN dateel.t_stock_sto.sto_are_code IS 'Code of the area, areas are geographic
see tr_area_are.';
COMMENT ON COLUMN dateel.t_stock_sto.sto_cou_code IS 'Code of the country see tr_country_cou
COMMENT ON COLUMN dateel.t_stock_sto.sto_lfs_code IS 'Code of the lifestage see tr_lifestage
both lfs_code, and lfs_spe_code (as two species can have the same lifestage code.';
COMMENT ON COLUMN dateel.t_stock_sto.sto_hty_code IS 'Code of the habitat type, one of MO (m
T (Transitional water), FW (Freshwater), null accepted';
COMMENT ON COLUMN dateel.t_stock_sto.sto_fia_code IS 'For marine area, code of the ICES area
COMMENT ON COLUMN dateel.t_stock_sto.sto_qal_code IS 'Code of data quality (1 good quality,
3 bad quality (not used), 4 dubious, 18, 19 ... historical data not used.
Not null, Foreign key set to tr_quality_qal';
COMMENT ON COLUMN dateel.t_stock_sto.sto_qal_comment IS 'Comment for the quality, for instan
COMMENT ON COLUMN dateel.t_stock_sto.sto_comment IS 'Comment on the value';
COMMENT ON COLUMN dateel.t_stock_sto.sto_datelastupdate IS 'Last update of the data';
COMMENT ON COLUMN dateel.t_stock_sto.sto_mis_code IS 'When no value are given in sto_value,
references table tr_missvalueqal_mis, should be null if value is provided (can''t have both)
COMMENT ON COLUMN dateel.t_stock_sto.sto_dta_code IS 'Access to data, default is ''Public'''
COMMENT ON COLUMN dateel.t_stock_sto.sto_wkg_code IS 'Code of the working group, one of
WGBAST, WGEEL, WGNAS, WKTRUTTA';
COMMENT ON COLUMN dateel.t_stock_sto.sto_ver_code IS 'Version code, references refeel. tr_ve

-- trigger on date
DROP FUNCTION IF EXISTS dateel.update_sto_datelastupdate CASCADE;
CREATE OR REPLACE FUNCTION dateel.update_sto_datelastupdate()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN
    NEW.sto_datelastupdate = now()::date;
    RETURN NEW;
END;
$function$
;
ALTER FUNCTION dateel.update_sto_datelastupdate() OWNER TO diaspara_admin;


CREATE TRIGGER update_sto_datelastupdate BEFORE
INSERT
    OR
UPDATE
    ON
    dateel.t_stock_sto FOR EACH ROW EXECUTE FUNCTION dateel.update_sto_datelastupdate();
```

SQL code to insert values in table `dateel.t_stock_sto`

```
DELETE FROM dateel.t_stock_sto;
INSERT INTO dateel.t_stock_sto
(sto_id, sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code,
sto_cou_code, sto_lfs_code, sto_hty_code,  sto_qal_code,
sto_qal_comment, sto_comment, sto_datelastupdate, sto_mis_code,
sto_dta_code, sto_wkg_code, sto_ver_code)
SELECT
eel_id AS  sto_id
, m.met_var AS sto_met_var
, e.eel_year AS sto_year
, '127186' AS  sto_spe_code
, e.eel_value AS sto_value
, CASE WHEN e.eel_emu_nameshort ilike '%total' THEN eel_cou_code
       WHEN e.eel_emu_nameshort IS NULL THEN eel_cou_code
ELSE e.eel_emu_nameshort END AS sto_are_code
, e.eel_cou_code AS sto_cou_code
, e.eel_lfs_code  AS sto_lfs_code
, CASE
      WHEN e.eel_hty_code = 'AL' THEN NULL
      WHEN e.eel_hty_code = 'F' THEN 'FW'
      WHEN e.eel_hty_code = 'MO' THEN 'MO'
      WHEN e.eel_hty_code = 'C' THEN 'MC'
      WHEN e.eel_hty_code = 'T' THEN 'T'
      WHEN e.eel_hty_code IS NULL THEN NULL
      ELSE 'TROUBLE' END AS sto_hty_code
--, NULL AS sto_fia_code -- fishing area
, e.eel_qal_id AS sto_qal_code -- see later TO INSERT deprecated values
, e.eel_qal_comment AS sto_qal_comment
, e.eel_comment AS sto_comment
, e.eel_datelastupdate AS sto_datelastupdate
, e.eel_missvaluequal AS sto_mis_code
, 'Public' AS sto_dta_code
, 'WGEEL' AS sto_wkg_code
, CASE
     WHEN e.eel_datasource = 'wgeel_2016' THEN 'WGEEL-2016-1'
     WHEN e.eel_datasource = 'dc_2017' THEN 'WGEEL-2017-1'
     WHEN e.eel_datasource = 'weel_2017' THEN 'WGEEL-2017-2'
     WHEN e.eel_datasource = 'dc_2018' THEN 'WGEEL-2018-1'
```

```
     WHEN e.eel_datasource = 'dc_2019' THEN 'WGEEL-2019-1'
     WHEN e.eel_datasource = 'dc_2020' THEN 'WGEEL-2020-1'
     WHEN e.eel_datasource = 'dc_2021' THEN 'WGEEL-2021-1'
     WHEN e.eel_datasource = 'dc_2022' THEN 'WGEEL-2022-1'
     WHEN e.eel_datasource = 'dc_2023' THEN 'WGEEL-2023-1'
     WHEN e.eel_datasource = 'dc_2024' THEN 'WGEEL-2024-1'
     WHEN e.eel_datasource = 'wkemp_2025' THEN 'WGEEL-2025-1'
     ELSE 'TROUBLE AND THIS SHOULD FAIL' END AS sto_ver_code
FROM datwgeel.t_eelstock_eel e
JOIN dateel.t_metadata_met m ON m.met_type::int = e.eel_typ_id
WHERE eel_qal_id IN (0,1,2,3,4)
AND eel_hty_code NOT IN ('T','M','C')
AND eel_missvaluequal != 'ND'
AND eel_typ_id != 16 -- habitat surface
; -- 28247

/*
SELECT distinct eel_datasource FROM datwgeel.t_eelstock_eel as t
SELECT distinct eel_hty_code FROM datwgeel.t_eelstock_eel
SELECT * FROM  datwgeel.t_eelstock_eel
WHERE eel_missvaluequal = 'ND'
AND eel_qal_id IN (0,1,2,3,4); --123 lines not kept
*/


-- OK I can remove for all T


INSERT INTO dateel.t_stock_sto
(sto_id, sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code,
sto_cou_code, sto_lfs_code, sto_hty_code,  sto_qal_code,
sto_qal_comment, sto_comment, sto_datelastupdate, sto_mis_code,
sto_dta_code, sto_wkg_code, sto_ver_code)
SELECT
eel_id AS sto_id
, m.met_var AS sto_met_var
, e.eel_year AS sto_year
, '127186' AS  sto_spe_code
, e.eel_value AS sto_value
, CASE WHEN e.eel_emu_nameshort ilike '%total' THEN eel_cou_code
       WHEN e.eel_emu_nameshort IS NULL THEN eel_cou_code
ELSE e.eel_emu_nameshort END AS sto_are_code
, e.eel_cou_code AS sto_cou_code
, e.eel_lfs_code  AS sto_lfs_code
, CASE
```

```
      WHEN e.eel_hty_code = 'AL' THEN NULL
      WHEN e.eel_hty_code = 'F' THEN 'FW'
      WHEN e.eel_hty_code = 'MO' THEN 'MO'
      WHEN e.eel_hty_code = 'C' THEN 'MC'
      WHEN e.eel_hty_code = 'T' THEN 'T'
      WHEN e.eel_hty_code IS NULL THEN NULL
      ELSE 'TROUBLE' END AS sto_hty_code
--, NULL AS sto_fia_code -- fishing area
, e.eel_qal_id AS sto_qal_code -- see later TO INSERT deprecated values
, e.eel_qal_comment AS sto_qal_comment
, e.eel_comment AS sto_comment
, e.eel_datelastupdate AS sto_datelastupdate
, e.eel_missvaluequal AS sto_mis_code
, 'Public' AS sto_dta_code
, 'WGEEL' AS sto_wkg_code
, CASE
     WHEN e.eel_datasource = 'wgeel_2016' THEN 'WGEEL-2016-1'
     WHEN e.eel_datasource = 'dc_2017' THEN 'WGEEL-2017-1'
     WHEN e.eel_datasource = 'weel_2017' THEN 'WGEEL-2017-2'
     WHEN e.eel_datasource = 'dc_2018' THEN 'WGEEL-2018-1'
     WHEN e.eel_datasource = 'dc_2019' THEN 'WGEEL-2019-1'
     WHEN e.eel_datasource = 'dc_2020' THEN 'WGEEL-2020-1'
     WHEN e.eel_datasource = 'dc_2021' THEN 'WGEEL-2021-1'
     WHEN e.eel_datasource = 'dc_2022' THEN 'WGEEL-2022-1'
     WHEN e.eel_datasource = 'dc_2023' THEN 'WGEEL-2023-1'
     WHEN e.eel_datasource = 'dc_2024' THEN 'WGEEL-2024-1'
     WHEN e.eel_datasource = 'wkemp_2025' THEN 'WGEEL-2025-1'
     ELSE 'TROUBLE AND THIS SHOULD FAIL' END AS sto_ver_code
FROM datwgeel.t_eelstock_eel e
JOIN dateel.t_metadata_met m ON m.met_type::int = e.eel_typ_id
WHERE eel_qal_id IN (0,1,2,3,4)
AND eel_hty_code IN ('T')
AND  eel_typ_id != 16
AND eel_missvaluequal != 'ND' ; -- 12303



-------------------------------------------------------------------------------------------------
 -- Do we have MO data ?
 -------------------------------------------------------------------------------------------------
SELECT * FROM  datwgeel.t_eelstock_eel  WHERE eel_hty_code = 'MO';--15600
SELECT * FROM  datwgeel.t_eelstock_eel  WHERE eel_hty_code = 'MO' and eel_value is not NULL
AND eel_qal_id in (1,2,3,4) ;
--26 (typ 8 and 9 there are releases for the rhone, no marine division.
-- So except for these 26 lines all data are MC.
-- I asked to Guirec, he will fix this in the database. So there should no longer be any mar
```

```
-- this is only for some data in MO (FR_Rhon) I'm inserting it now
SELECT *
FROM datwgeel.t_eelstock_eel e
JOIN dateel.t_metadata_met m ON m.met_type::int = e.eel_typ_id
WHERE eel_qal_id IN (0,1,2,3,4)
AND eel_hty_code IN ('MO')
AND eel_value is not NULL
AND eel_missvaluequal != 'ND';

INSERT INTO dateel.t_stock_sto
(sto_id, sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code,
sto_cou_code, sto_lfs_code, sto_hty_code, sto_qal_code,
sto_qal_comment, sto_comment, sto_datelastupdate, sto_mis_code,
sto_dta_code, sto_wkg_code, sto_ver_code)
SELECT
eel_id AS sto_id
, m.met_var AS sto_met_var
, e.eel_year AS sto_year
, '127186' AS  sto_spe_code
, e.eel_value AS sto_value
, CASE WHEN eel_emu_nameshort = 'FR_Rhon' THEN '37.1.2.7'
  ELSE 'STOP' END AS sto_are_code
, e.eel_cou_code AS sto_cou_code
, e.eel_lfs_code  AS sto_lfs_code
, CASE
      WHEN e.eel_hty_code = 'AL' THEN NULL
      WHEN e.eel_hty_code = 'F' THEN 'FW'
      WHEN e.eel_hty_code = 'MO' THEN 'MO'
      WHEN e.eel_hty_code = 'C' THEN 'MC'
      WHEN e.eel_hty_code = 'T' THEN 'T'
      WHEN e.eel_hty_code IS NULL THEN NULL
      ELSE 'TROUBLE' END AS sto_hty_code
--, NULL AS sto_fia_code -- fishing area
, e.eel_qal_id AS sto_qal_code -- see later TO INSERT deprecated values
, e.eel_qal_comment AS sto_qal_comment
, e.eel_comment AS sto_comment
, e.eel_datelastupdate AS sto_datelastupdate
, e.eel_missvaluequal AS sto_mis_code
, 'Public' AS sto_dta_code
, 'WGEEL' AS sto_wkg_code
, CASE
     WHEN e.eel_datasource = 'wgeel_2016' THEN 'WGEEL-2016-1'
     WHEN e.eel_datasource = 'dc_2017' THEN 'WGEEL-2017-1'
     WHEN e.eel_datasource = 'weel_2017' THEN 'WGEEL-2017-2'
```

```
     WHEN e.eel_datasource = 'dc_2018' THEN 'WGEEL-2018-1'
     WHEN e.eel_datasource = 'dc_2019' THEN 'WGEEL-2019-1'
     WHEN e.eel_datasource = 'dc_2020' THEN 'WGEEL-2020-1'
     WHEN e.eel_datasource = 'dc_2021' THEN 'WGEEL-2021-1'
     WHEN e.eel_datasource = 'dc_2022' THEN 'WGEEL-2022-1'
     WHEN e.eel_datasource = 'dc_2023' THEN 'WGEEL-2023-1'
     WHEN e.eel_datasource = 'dc_2024' THEN 'WGEEL-2024-1'
     WHEN e.eel_datasource = 'wkemp_2025' THEN 'WGEEL-2025-1'
     ELSE 'TROUBLE AND THIS SHOULD FAIL' END AS sto_ver_code
FROM datwgeel.t_eelstock_eel e
JOIN dateel.t_metadata_met m ON m.met_type::int = e.eel_typ_id
WHERE eel_qal_id IN (0,1,2,3,4)
AND eel_hty_code IN ('MO')
AND eel_value is not NULL; --26




-------------------------------------------------------------------------------------------
-- Coastal waters, in which case do we have more than one eel_area_division for one emu, one
-------------------------------------------------------------------------------------------


with dupl AS (
SELECT *, count(eel_area_division)  OVER (PARTITION BY eel_typ_id, eel_emu_nameshort,  eel_y
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_hty_code = 'C'
  AND eel_value is not NULL
  AND eel_qal_id in (1,2,3,4))
  SELECT * FROM dupl WHERE count>1; --98
-- OK so I have landings for SE_East, DK_Mari, and NO_total. In all those cases I can use th

-------------------------------------------------------------------------------------------
-- Coastal waters, in which case do we have one value per eel_area_division for one emu, one
-- Are there countries where more than one EMU is reported ?
-------------------------------------------------------------------------------------------

with dupl AS (
SELECT *, count(eel_area_division)  OVER (PARTITION BY eel_typ_id, eel_emu_nameshort,  eel_y
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_hty_code = 'C'
  AND eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_typ_id !=16)
  SELECT * FROM dupl WHERE count=1
```

```
  ORDER BY eel_emu_nameshort, eel_typ_id;


  -- BE_Sche is 27.4.c typ_id 6 (rec)
  -- DE_Eide is always 27.4.b, typ_id 4 (com) and 6 (rec)
  -- DE_Schl is always 27.3.b, c, typ id 4 6 and 9 (release OG and G)
  -- => DE_Warn is always 27.3.d except for habitat where it is reported as 27.3.b,c o I'm r
  -- DK_Inla  1 value in 2021 eel_id 569486, typ_id 4 Y dc_2024 => removed below
SELECT * FROM datwgeel.t_eelstock_eel
  WHERE
  eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_typ_id = 4
  AND eel_cou_code = 'DK'
  AND eel_lfs_code = 'Y'


  -- Dk_Mari is always 27.3.b, c, Y or S, typ_id 4

  -- DK_total is always 27.3.b, c typ_id 6, eel_lfs (Y, YS) , 2017-2022
  -- EE_West is always 27.3.d, typ_id 4 (com) YS
  -- ES_Murc is always 37.1.1, typ_id 4 (com) YS
  -- !! ES_Vale On value G in, typ_id 4 in 2021 dc_2021 (OK should be T)

    SELECT *
 FROM  datwgeel.t_eelstock_eel
  WHERE
  eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_typ_id = 4
  AND eel_emu_nameshort = 'ES_Vale'
  AND eel_lfs_code = 'G'

  -- FI_Finl is always 27.3.d typ_id 4, 6, 9

  -- GR_EaMT is always 37.3.1
  -- GR_NorW is always 37.2.2
  -- GR_WePe is always 37.2.2
  -- !! GR_total is reporting both with an without eel_area_division for GR_total => Mail se
  -- LT_Lith / Lt_total change an make it consistent ?
  -- LV_latv is always  37.3.d  4, 6
  -- !! NL_Neth is always 27.4.c except for two lines where I have nulls, mail sent for chec
  -- !! NO_Total is always 27.7.a (wrong)
  -- except for 3 yellow lines in 2021-2023 where it becomes 37.3.a => mail sent
  -- PL_Oder and PL_Vist are all 37.3.d 4,8,9
```

```
  -- !! SE East is always reporting 27.3.d, 27.3b,c (Baltic this is consistent with emu_def
  -- SE WE  is always reporting 27.3.a
  -- Sl_total is always reporting 37.2.1
  -- TN_EC is always reporting 37.2.2
  -- TN_NE is always reporting 37.1.3
  -- TN_SO is always reporting 37.2.2
  -- Change for tunisia ? Should be Inland for the lagoons...;

-- so Greece, Poland are reporting two rows with different EMUs. For Greece and Poland make
-- Tunisia is also reporting more than one EMU for 37.2.2 but this



-----------------------------------------------------
-- Coastal water not reported with eel_area_division
-----------------------------------------------------
SELECT *
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_hty_code = 'C'
  AND eel_area_division IS NULL
  AND eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_typ_id !=16
   ORDER BY eel_emu_nameshort, eel_typ_id, eel_lfs_code, eel_year; --1061




-- DE_Eide 8 G 2020-2022 (value 0)
-- DE_Eide 8 OG 1985-2022 (value 0)
-- DE_Eide 8 Y 1985-2022 (value 0)
-- DE_Eide 9 (same)  should be  27.4.b
-- DE_Schl 1985-2022 OG 0 and then values => Should be  27.3.b, c
-- DE_Warn should be 27.3.d

-- DK Mari 2016 2024 8 - 9 OG

SELECT *
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_typ_id IN (8,9)
  AND eel_cou_code = 'DK'
   ORDER BY eel_emu_nameshort, eel_typ_id, eel_lfs_code, eel_year;
```

```
-- DK total says ICES subdivision 21 22 23 24 (not tin the list) this comment shows that div
-- Mail sent to Michael 25/07 for check...

-- EE_West should always be  27.3.d
-- missing values for 6 lines (4, 6) Mail sent to Paul
SELECT *

 FROM  datwgeel.t_eelstock_eel
  WHERE eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_cou_code = 'EE'
   ORDER BY eel_emu_nameshort, eel_typ_id, eel_lfs_code, eel_year;

 SELECT * FROM  datwgeel.t_eelstock_eel
  WHERE eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_cou_code = 'EE'
  AND eel_hty_code IS NOT NULL
  AND eel_typ_id= 11
  ORDER BY eel_emu_nameshort, eel_typ_id, eel_lfs_code, eel_year;

-- remove duplicates for aquaculture (2002 to 2016) some qal_id 3, in fact data are entered

-- SE Mail sent to Josefin and Rob

SELECT * FROM
 datwgeel.t_eelstock_eel
  WHERE eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_hty_code = 'C'
  AND eel_emu_nameshort = 'SE_total'



-- Check transitional waters 4, 6 in 2019 and 4 2020 2023 YS




-- see database_edition_2025.sql in wgeel for query
```

```
with dupl AS (
SELECT *, count(eel_area_division)  OVER (PARTITION BY eel_typ_id, eel_emu_nameshort,  eel_y
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_hty_code = 'T'
  AND eel_value is not NULL
  AND eel_qal_id in (1,2,3,4))
  SELECT * FROM dupl WHERE count>1; --2

-- OK there is a duplicate with qal_id 3 for ES_Cata ES G T 37.1.1

with dupl AS (
SELECT *, count(eel_area_division)  OVER (PARTITION BY eel_typ_id, eel_emu_nameshort,  eel_y
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_hty_code = 'T'
  AND eel_value is not NULL
  AND eel_qal_id in (1,2,3,4))
  SELECT * FROM dupl WHERE count=1; --1254

with dupl AS (
SELECT *, count(eel_area_division)  OVER (PARTITION BY eel_typ_id, eel_emu_nameshort,  eel_y
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_hty_code = 'T'
  AND eel_value is not NULL
  AND eel_qal_id in (1,2,3,4))
  SELECT DISTINCT ON (eel_typ_id, eel_emu_nameshort, eel_lfs_code, eel_area_division) eel_ty
 ; --1254

-----------------------------------------------------------------
-- insert coastal waters where eel_area_division is not NULL
 -----------------------------------------------------------------

 DROP TABLE tempo.emu_div;
 CREATE TABLE tempo.emu_div AS
 SELECT DISTINCT ON (eel_cou_code, eel_emu_nameshort, eel_area_division)
 eel_cou_code, eel_emu_nameshort,
  eel_area_division,
  NULL AS area_code
 FROM  datwgeel.t_eelstock_eel
  WHERE eel_value is not NULL
  AND eel_qal_id in (1,2,3,4)
  AND eel_area_division IS NOT NULL
  order by (eel_emu_nameshort);--21
  UPDATE tempo.emu_div set area_code = eel_area_division
```

```
  WHERE eel_emu_nameshort IN
  (SELECT emu_nameshort from refwgeel.tr_emusplit_ems where emu_sea like '%A%' OR emu_sea =


  SELECT * FROM datwgeel.t_eelstock_eel WHERE eel_emu_nameshort='IT_Pugl' AND eel_area_divis
   SELECT * FROM datwgeel.t_eelstock_eel WHERE eel_emu_nameshort='NO_total' AND eel_area_div
DELETE FROM tempo.emu_div WHERE eel_emu_nameshort='IT_Pugl' AND eel_area_division='37.2.2';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='AL_total' AND eel_area_division='37.2.2';
UPDATE tempo.emu_div
  SET area_code='27.3.b, c'
  WHERE eel_emu_nameshort='DK_total' AND eel_area_division='27.3.b, c';
UPDATE tempo.emu_div
  SET area_code='37.1.1.4'
  WHERE eel_emu_nameshort='DZ_total' AND eel_area_division='37.1.1';
UPDATE tempo.emu_div
  SET area_code='37.3.2.26'
  WHERE eel_emu_nameshort='EG_total' AND eel_area_division='37.3.2';
UPDATE tempo.emu_div
  SET area_code='37.3.1.22'
  WHERE eel_emu_nameshort='GR_EaMT' AND eel_area_division='37.3.1';
UPDATE tempo.emu_div
  SET area_code='37.2.2.20'
  WHERE eel_emu_nameshort='GR_NorW' AND eel_area_division='37.2.2';
UPDATE tempo.emu_div
  SET area_code='37.2.2.20'
  WHERE eel_emu_nameshort='GR_WePe' AND eel_area_division='37.2.2';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='HR_total' AND eel_area_division='37.2.1';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='IT_Emil' AND eel_area_division='37.2.1';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='IT_Frio' AND eel_area_division='37.2.1';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='IT_Lazi' AND eel_area_division='37.1.3';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='IT_Pugl' AND eel_area_division='37.2.1';
UPDATE tempo.emu_div
  SET area_code='37.3.1.112'
```

```
  WHERE eel_emu_nameshort='IT_Sard' AND eel_area_division='37.1.3';
UPDATE tempo.emu_div
  SET area_code='37.1.3.9'
  WHERE eel_emu_nameshort='IT_Tosc' AND eel_area_division='37.1.3';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='IT_Vene' AND eel_area_division='37.2.1';
UPDATE tempo.emu_div
  SET area_code='37.2.1.17'
  WHERE eel_emu_nameshort='SI_total' AND eel_area_division='37.2.1';
UPDATE tempo.emu_div
  SET area_code='37.2.2.13'
  WHERE eel_emu_nameshort='TN_EC' AND eel_area_division='37.2.2';
UPDATE tempo.emu_div
  SET area_code='37.1.3.12'
  WHERE eel_emu_nameshort='TN_NE' AND eel_area_division='37.1.3';
UPDATE tempo.emu_div
  SET area_code='37.1.3.12'
  WHERE eel_emu_nameshort='TN_Nor' AND eel_area_division='37.1.3';
UPDATE tempo.emu_div
  SET area_code='37.2.2.14'
  WHERE eel_emu_nameshort='TN_SO' AND eel_area_division='37.2.2';

DELETE FROM tempo.emu_div
  WHERE eel_emu_nameshort='ES_Mino' AND eel_area_division='37.1.1';
DELETE FROM tempo.emu_div
  WHERE eel_emu_nameshort='ES_Vale' AND eel_area_division='37.1.2';
DELETE FROM tempo.emu_div
  WHERE eel_emu_nameshort='IT_Pugl' AND eel_area_division='37.2.2';
UPDATE tempo.emu_div
  SET area_code='37.1.1.5'
  WHERE eel_emu_nameshort='ES_Bale' AND eel_area_division='37.1.1';
UPDATE tempo.emu_div
  SET area_code='37.1.1.6'
  WHERE eel_emu_nameshort='ES_Cata' AND eel_area_division='37.1.1';
UPDATE tempo.emu_div
  SET area_code='27.9.a'
  WHERE eel_emu_nameshort='ES_Minh' AND eel_area_division='27.9.a';
UPDATE tempo.emu_div
  SET area_code='27.9.a'
  WHERE eel_emu_nameshort='ES_Mino' AND eel_area_division='27.9.a';
UPDATE tempo.emu_div
  SET area_code='37.1.1.6'
  WHERE eel_emu_nameshort='ES_Vale' AND eel_area_division='37.1.1';
UPDATE tempo.emu_div
```

```
  SET area_code='37.1.1.1'
  WHERE eel_emu_nameshort='ES_Murc' ;
UPDATE tempo.emu_div
  SET area_code='27.3.d'
  WHERE eel_emu_nameshort='LT_total' AND eel_area_division='27.3.d';
UPDATE tempo.emu_div
  SET area_code='27.3.a'
  WHERE eel_emu_nameshort='NO_total' AND eel_area_division='27.3.a';
UPDATE tempo.emu_div
  SET area_code='27.4.a'
  WHERE eel_emu_nameshort='NO_total' AND eel_area_division='27.4.a';
UPDATE tempo.emu_div
  SET area_code='27.2.a'
  WHERE eel_emu_nameshort='NO_total' AND eel_area_division='27.7.a';
UPDATE tempo.emu_div
  SET area_code='27.3.d'
  WHERE eel_emu_nameshort='PL_total' AND eel_area_division='27.3.d';
UPDATE tempo.emu_div
  SET eel_area_division='27.9.a',area_code='27.9.a'
  WHERE eel_emu_nameshort='PT_total' AND eel_area_division='27.9.a';
UPDATE tempo.emu_div
  SET area_code='27.3.d'
  WHERE eel_emu_nameshort='SE_Ea_o' AND eel_area_division='27.3.d';
UPDATE tempo.emu_div
  SET area_code='27.3.d'
  WHERE eel_emu_nameshort='SE_So_o' AND eel_area_division='27.3.d';
UPDATE tempo.emu_div
  SET area_code='37.3.1.22'
  WHERE eel_emu_nameshort='GR_total' AND eel_area_division='37.3.1' ;









INSERT INTO tempo.emu_div (eel_cou_code,eel_emu_nameshort,area_code)
  VALUES ('EE','EE_West','27.3.d')
INSERT INTO tempo.emu_div (eel_cou_code,eel_emu_nameshort,area_code)
  VALUES ('EE','EE_West','27.3.d')

INSERT INTO dateel.t_stock_sto
(sto_id, sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code,
```

```
sto_cou_code, sto_lfs_code, sto_hty_code,  sto_qal_code,
sto_qal_comment, sto_comment, sto_datelastupdate, sto_mis_code,
sto_dta_code, sto_wkg_code, sto_ver_code)
SELECT
eel_id AS sto_id
, m.met_var AS sto_met_var
, e.eel_year AS sto_year
, '127186' AS  sto_spe_code
, e.eel_value AS sto_value
, emu_div.area_code AS sto_are_code
--, e.eel_area_division
, e.eel_cou_code AS sto_cou_code
, e.eel_lfs_code  AS sto_lfs_code
, CASE
      WHEN e.eel_hty_code = 'AL' THEN NULL
      WHEN e.eel_hty_code = 'F' THEN 'FW'
      WHEN e.eel_hty_code = 'MO' THEN 'MO'
      WHEN e.eel_hty_code = 'C' THEN 'MC'
      WHEN e.eel_hty_code = 'T' THEN 'T'
      WHEN e.eel_hty_code IS NULL THEN NULL
      ELSE 'TROUBLE' END AS sto_hty_code
--, NULL AS sto_fia_code -- fishing area
, e.eel_qal_id AS sto_qal_code
, e.eel_qal_comment AS sto_qal_comment
, e.eel_comment AS sto_comment
, e.eel_datelastupdate AS sto_datelastupdate
, e.eel_missvaluequal AS sto_mis_code
, 'Public' AS sto_dta_code
, 'WGEEL' AS sto_wkg_code
, CASE
     WHEN e.eel_datasource = 'wgeel_2016' THEN 'WGEEL-2016-1'
     WHEN e.eel_datasource = 'dc_2017' THEN 'WGEEL-2017-1'
     WHEN e.eel_datasource = 'weel_2017' THEN 'WGEEL-2017-2'
     WHEN e.eel_datasource = 'dc_2018' THEN 'WGEEL-2018-1'
     WHEN e.eel_datasource = 'dc_2019' THEN 'WGEEL-2019-1'
     WHEN e.eel_datasource = 'dc_2020' THEN 'WGEEL-2020-1'
     WHEN e.eel_datasource = 'dc_2021' THEN 'WGEEL-2021-1'
     WHEN e.eel_datasource = 'dc_2022' THEN 'WGEEL-2022-1'
     WHEN e.eel_datasource = 'dc_2023' THEN 'WGEEL-2023-1'
     WHEN e.eel_datasource = 'dc_2024' THEN 'WGEEL-2024-1'
     WHEN e.eel_datasource = 'wkemp_2025' THEN 'WGEEL-2025-1'
     ELSE 'TROUBLE AND THIS SHOULD FAIL' END AS sto_ver_code
FROM datwgeel.t_eelstock_eel e
JOIN dateel.t_metadata_met m ON m.met_type::int = e.eel_typ_id
LEFT JOIN tempo.emu_div ON (emu_div.eel_emu_nameshort,emu_div.eel_area_division) = (e.eel_em
```

```
WHERE eel_qal_id IN (0,1,2,3,4)
AND e.eel_hty_code ='C'
AND e.eel_value IS NOT NULL
AND e.eel_area_division IS NOT NULL; -- 1295

-------------------------------------------------------
-- insert coastal waters where eel_area_division is NULL
-------------------------------------------------------

CREATE TABLE tempo.emu_null AS SELECT * FROM tempo.emu_div WHERE
eel_emu_nameshort IN (SELECT DISTINCT eel_emu_nameshort FROM
datwgeel.t_eelstock_eel e
WHERE e.eel_qal_id IN (0,1,2,3,4)
AND e.eel_hty_code ='C'
AND e.eel_value IS NOT NULL
AND e.eel_area_division IS NULL)

-- See joplin, I have to make some choice, the best for SE_East, DK, and NO would be to spli
-- and not have any null values.
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='DK_Mari' AND eel_area_division='27.3.a';
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='DK_Mari' AND eel_area_division='27.4.b';
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='DE_Warn' AND eel_area_division='27.3.b, c';
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='SE_East' AND eel_area_division='27.3.a';
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='SE_East' AND eel_area_division='27.3.b, c';
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='NO_total' AND eel_area_division='27.4.a';
DELETE FROM tempo.emu_null
  WHERE eel_emu_nameshort='NO_total' AND eel_area_division='27.7.a';



INSERT INTO dateel.t_stock_sto
(sto_id, sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code,
sto_cou_code, sto_lfs_code, sto_hty_code,  sto_qal_code,
sto_qal_comment, sto_comment, sto_datelastupdate, sto_mis_code,
sto_dta_code, sto_wkg_code, sto_ver_code)
SELECT
eel_id AS sto_id
, m.met_var AS sto_met_var
, e.eel_year AS sto_year
```

```
, '127186' AS  sto_spe_code
, e.eel_value AS sto_value
, emu_null.area_code AS sto_are_code
--, e.eel_area_division
--, e.eel_emu_nameshort
, e.eel_cou_code AS sto_cou_code
, e.eel_lfs_code  AS sto_lfs_code
, CASE
      WHEN e.eel_hty_code = 'AL' THEN NULL
      WHEN e.eel_hty_code = 'F' THEN 'FW'
      WHEN e.eel_hty_code = 'MO' THEN 'MO'
      WHEN e.eel_hty_code = 'C' THEN 'MC'
      WHEN e.eel_hty_code = 'T' THEN 'T'
      WHEN e.eel_hty_code IS NULL THEN NULL
      ELSE 'TROUBLE' END AS sto_hty_code
--, NULL AS sto_fia_code -- fishing area
, e.eel_qal_id AS sto_qal_code
, e.eel_qal_comment AS sto_qal_comment
, e.eel_comment AS sto_comment
, e.eel_datelastupdate AS sto_datelastupdate
, e.eel_missvaluequal AS sto_mis_code
, 'Public' AS sto_dta_code
, 'WGEEL' AS sto_wkg_code
, CASE
     WHEN e.eel_datasource = 'wgeel_2016' THEN 'WGEEL-2016-1'
     WHEN e.eel_datasource = 'dc_2017' THEN 'WGEEL-2017-1'
     WHEN e.eel_datasource = 'weel_2017' THEN 'WGEEL-2017-2'
     WHEN e.eel_datasource = 'dc_2018' THEN 'WGEEL-2018-1'
     WHEN e.eel_datasource = 'dc_2019' THEN 'WGEEL-2019-1'
     WHEN e.eel_datasource = 'dc_2020' THEN 'WGEEL-2020-1'
     WHEN e.eel_datasource = 'dc_2021' THEN 'WGEEL-2021-1'
     WHEN e.eel_datasource = 'dc_2022' THEN 'WGEEL-2022-1'
     WHEN e.eel_datasource = 'dc_2023' THEN 'WGEEL-2023-1'
     WHEN e.eel_datasource = 'dc_2024' THEN 'WGEEL-2024-1'
     WHEN e.eel_datasource = 'wkemp_2025' THEN 'WGEEL-2025-1'
     ELSE 'TROUBLE AND THIS SHOULD FAIL' END AS sto_ver_code
FROM datwgeel.t_eelstock_eel e
JOIN dateel.t_metadata_met m ON m.met_type::int = e.eel_typ_id
LEFT JOIN tempo.emu_null ON (emu_null.eel_emu_nameshort) = (e.eel_emu_nameshort)
WHERE eel_qal_id IN (0,1,2,3,4)
AND e.eel_hty_code ='C'
AND e.eel_value IS NOT NULL
AND e.eel_area_division IS NULL
AND e. eel_emu_nameshort != 'SE_total'; -- 1057
```

```
SELECT * FROM datwgeel.t_eelstock_eel WHERE eel_id ='567218';

SELECT * FROM datwgeel.t_eelstock_eel WHERE
eel_qal_id IN (0,1,2,3,4)
AND eel_hty_code ='C'
AND eel_value IS NOT NULL
AND eel_area_division IS NULL
AND eel_emu_nameshort = 'SE_total'; --4 lines


-- TODO eel_percent
-- TODO see later TO INSERT deprecated values
```

Chapter 6

WGBAST

SQL code to create table `datbast.t_metadata_met`

```
DROP TABLE IF EXISTS datbast.t_metadata_met CASCADE;

CREATE TABLE datbast.t_metadata_met(
 CONSTRAINT t_metadata_met_pkey PRIMARY KEY(met_var, met_spe_code),
 CONSTRAINT fk_met_spe_code FOREIGN KEY (met_spe_code)
  REFERENCES ref.tr_species_spe(spe_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
 CONSTRAINT ck_met_spe_code CHECK
 (met_spe_code='127186' OR met_spe_code='127187'),
 CONSTRAINT fk_met_wkg_code FOREIGN KEY (met_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT ck_met_wkg_code CHECK (met_wkg_code='WGBAST'),
  CONSTRAINT fk_met_ver_code FOREIGN KEY (met_ver_code)
  REFERENCES refbast.tr_version_ver(ver_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_oty_code FOREIGN KEY (met_oty_code)
  REFERENCES ref.tr_objecttype_oty (oty_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_nim_code FOREIGN KEY (met_nim_code)
  REFERENCES ref.tr_nimble_nim (nim_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_mtr_code FOREIGN KEY (met_mtr_code)
  REFERENCES ref.tr_metric_mtr(mtr_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_uni_code FOREIGN KEY (met_uni_code)
  REFERENCES ref.tr_units_uni(uni_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
```

```
  CONSTRAINT fk_met_cat_code FOREIGN KEY (met_cat_code)
  REFERENCES ref.tr_category_cat(cat_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_met_des_code FOREIGN KEY (met_des_code)
  REFERENCES ref.tr_destination_des(des_code)
  ON UPDATE CASCADE ON DELETE RESTRICT
)
INHERITS (dat.t_metadata_met);


--  COMMENTS FOR WGEEL

COMMENT ON TABLE datbast.t_metadata_met IS
'Table (metadata) of each variable (parameter) in the wgeel database.';
COMMENT ON COLUMN datbast.t_metadata_met.met_var
IS 'Variable code, primary key on both met_spe_code and met_var.';
COMMENT ON COLUMN datbast.t_metadata_met.met_spe_code
IS 'Species aphiaID, text ''127186'' salmo salar OR ''127187'' for Salmo trutta primary key
COMMENT ON COLUMN datbast.t_metadata_met.met_ver_code
IS 'Code on the version of the model, see table refeel.tr_version_ver.';
COMMENT ON COLUMN datbast.t_metadata_met.met_oty_code
IS 'Object type, single_value, vector, matrix see table tr_objecttype_oty.';
COMMENT ON COLUMN datbast.t_metadata_met.met_nim_code
IS 'Nimble type, one of data, constant, output, other.';
COMMENT ON COLUMN datbast.t_metadata_met.met_dim
IS 'Dimension of the Nimble variable, use {10, 100, 100}
to insert the description of an array(10,100,100).';
COMMENT ON COLUMN datbast.t_metadata_met.met_dimname
IS 'Dimension of the variable in Nimble, use {''year'', ''stage'', ''area''}.';
COMMENT ON COLUMN datbast.t_metadata_met.met_modelstage
IS 'Currently one of fit, other, First year.';
COMMENT ON COLUMN datbast.t_metadata_met.met_type
IS 'Type of data in the variable, homewatercatches, Initialisation first year,
abundance ....';
COMMENT ON COLUMN datbast.t_metadata_met.met_location
IS 'Describe process with geographical information';
COMMENT ON COLUMN datbast.t_metadata_met.met_fishery
IS 'Description of the fishery.';
COMMENT ON COLUMN datbast.t_metadata_met.met_des_code
IS 'Destination of the fish, e.g. Released (alive), Seal damage,
Removed (from the environment), references table tr_destination_des., this is currently only
so can be kept NULL';
COMMENT ON COLUMN datbast.t_metadata_met.met_uni_code
IS 'Unit, datbasterences table tr_unit_uni.';
COMMENT ON COLUMN datbast.t_metadata_met.met_cat_code
```

```
IS 'Broad category of data or parameter,
catch, effort, biomass, mortality, count ...datbasterences table tr_category_cat.';
COMMENT ON COLUMN datbast.t_metadata_met.met_mtr_code
IS 'Code of the metric, datbasterences tr_metric_mtr, Estimate, Bound, SD, CV ....';
COMMENT ON COLUMN datbast.t_metadata_met.met_definition
IS 'Definition of the metric.';
COMMENT ON COLUMN datbast.t_metadata_met.met_deprecated
IS'Is the variable still used ?';



ALTER TABLE datbast.t_metadata_met OWNER TO diaspara_admin;
GRANT SELECT ON datbast.t_metadata_met TO diaspara_read;
```

An analysis of the WGBAST dataset for landings shows that it could follow the structure of the main t_stock_sto table, here is the list of changes needed.

- **gear**. The gear must be added to the dimension of the t_stock_sto table, it is one dimension of the table.
- **Time period**. The data are not always reported by YEAR, unlike in eel or WGNAS. Other types of time reporting e.g. Month, Half of year, Quarter need to be added to the t_stock_sto table, since this table is inherited in postgres and aready has one more column (to store some extra dimension) is WGNAS when compared to WGEEL, we need to do the same and allow for three additional columns, one for gear, one for time period type and one for time period.
- The metadata will allow by a simple join to get back to F_type (stored in column met_type). It should also for a simple division according to the destination column, allowing to separate dead fish from the living ones.
- **Effort, Numbers and Weights**. The database will be in long format while in the current structure, Effort, Weights and Numbers are reported in separate columns.A simple query will bring back the original format.
- Effort is reported in geardays only for driftnet, longline and trapnet fisheries.
- TODO CHECK WITH maria about the unit (effort gearxdays)
- TODO Check ALV ALL in gears
- TODO Check 138 rows without f_type, can these all be attributed to COMM

6.1 Create referential for versions WGBAST

Creating the version referential for WGBAST

```
DROP TABLE IF EXISTS refbast.tr_version_ver CASCADE;
CREATE TABLE refbast.tr_version_ver() inherits (ref.tr_version_ver);

ALTER TABLE refbast.tr_version_ver ADD CONSTRAINT ver_code_pkey PRIMARY KEY (ver_code);
ALTER TABLE refbast.tr_version_ver ADD CONSTRAINT  fk_ver_spe_code FOREIGN KEY (ver_spe_code
REFERENCES ref.tr_species_spe(spe_code)
ON UPDATE CASCADE ON DELETE CASCADE;

COMMENT ON TABLE refbast.tr_version_ver
IS 'Table of data or variable version, essentially one datacall or advice, inherits ref.tr_v

COMMENT ON COLUMN refbast.tr_version_ver.ver_code
IS 'Version code, stockkey-year-version.';
COMMENT ON COLUMN refbast.tr_version_ver.ver_year
IS 'Year of assessement.';
COMMENT ON COLUMN refbast.tr_version_ver.ver_spe_code
IS 'Species code left NULL for WGBAST as the data call references several species';
COMMENT ON COLUMN refbast.tr_version_ver.ver_stockkeylabel
IS 'Ver_stockkeylabel e.g. ele.2737.nea.';
COMMENT ON COLUMN refbast.tr_version_ver.ver_datacalldoi
IS 'Data call DOI, find a way to retrieve that information
and update this comment';
COMMENT ON COLUMN refbast.tr_version_ver.ver_version
IS 'Version code corresponding to numbering of the versions';
COMMENT ON COLUMN refbast.tr_version_ver.ver_description
IS 'Description of the data call / version.';
GRANT ALL ON refbast.tr_version_ver TO diaspara_admin;
GRANT SELECT ON refbast.tr_version_ver TO diaspara_read;
```

```
tr_version_ver <- data.frame(
  ver_code = paste0("WGNAS-",2020:2024,"-1"),
  ver_year = 2020:2024,
  ver_spe_code = "127186",
  ver_wkg_code = "WGNAS",
  ver_datacalldoi=c(NA,NA,NA,NA,"https://doi.org/10.17895/ices.pub.25071005.v3"),
  ver_stockkeylabel =c("sal.neac.all"), # sugested by Hilaire.
  # TODO FIND other DOI (mail sent to ICES)
  ver_version=c(1,1,1,1,1), # TODO WGNAS check that there is just one version per year
  ver_description=c(NA,NA,NA,NA,NA)) # TODO WGNAS provide model description
```

```r
DBI::dbWriteTable(con_diaspara_admin, "temp_tr_version_ver", tr_version_ver,
                  overwrite = TRUE)
dbExecute(con_diaspara_admin, "INSERT INTO refnas.tr_version_ver(ver_code, ver_year, ver_spe
DBI::dbExecute(con_diaspara_admin, "DROP TABLE temp_tr_version_ver;")

tr_version_ver <- data.frame(
  ver_code = paste0("WGBAST-",2024:2025,"-1"),
  ver_year = 2024:2025,
  ver_spe_code = NA,
  ver_wkg_code = "WGBAST",
  ver_datacalldoi=c("https://doi.org/10.17895/ices.pub.25071005.v3","https://doi.org/10.1789
  ver_stockkeylabel =c("sal.27.22-31"),
  # TODO FIND other DOI (mail sent to ICES)
  ver_version=c(1,1), # TODO WGNAS check that there is just one version per year
  ver_description=c("Joint ICES Fisheries Data call for landings, discards, biological and e

DBI::dbWriteTable(con_diaspara_admin, "temp_tr_version_ver", tr_version_ver,
                  overwrite = TRUE)
dbExecute(con_diaspara_admin, "INSERT INTO refbast.tr_version_ver(ver_code, ver_year, ver_sp
DBI::dbExecute(con_diaspara_admin, "DROP TABLE temp_tr_version_ver;")
```

6.2 Create datbast.tr_estimationmethod_esm

Creating the estimation method referential for WGBAST

```sql
-- Table of estimation methods when databasis is Estimated

DROP TABLE IF EXISTS refbast.tr_estimationmethod_esm CASCADE;
CREATE  TABLE refbast.tr_estimationmethod_esm () inherits (ref.tr_estimationmethod_esm);
ALTER TABLE refbast.tr_estimationmethod_esm ALTER COLUMN esm_wkg_code SET DEFAULT 'WGBAST';
ALTER TABLE refbast.tr_estimationmethod_esm ADD CONSTRAINT uk_esm_code UNIQUE (esm_code);
COMMENT ON TABLE refbast.tr_estimationmethod_esm IS 'Table of table estimation method, provi
COMMENT ON COLUMN refbast.tr_estimationmethod_esm.esm_code IS 'Estimation method code';
COMMENT ON COLUMN refbast.tr_estimationmethod_esm.esm_description IS 'Estimation method  des
COMMENT ON COLUMN refbast.tr_estimationmethod_esm.esm_icesvalue IS 'Code (Key) of the Estima
COMMENT ON COLUMN refbast.tr_estimationmethod_esm.esm_icesguid IS 'UUID (guid) of ICES ';
COMMENT ON COLUMN refbast.tr_estimationmethod_esm.esm_icestablesource IS 'Source table in IC

GRANT ALL ON refbast.tr_estimationmethod_esm TO diaspara_admin;
GRANT SELECT ON refbast.tr_estimationmethod_esm TO diaspara_read;
```

```r
#
```

```
esm <- data.frame(
  esm_id=1:8,
  esm_code = paste0("SmoltEst",1:8),
  esm_description = c("Estimate of smolt production from complete count of smolts.",
                      "Sampling of smolts and estimate of total smolt run size.",
                      "Estimate of smolt run from parr production by relation developed in t
                      "Estimate of smolt run from parr production by relation developed in a
                      "Inference of smolt production from data derived from similar rivers i
                      "Estimate of smolt production from count of spawners.",
                      "Estimate of smolt production inferred from stocking of reared fish in
                      "Estimate of smolt production from salmon catch, exploitation and surv
  esm_icesvalue = NA,
  esm_icesguid = NA,
  esm_icestablesource =NA
)


DBI::dbWriteTable(con_diaspara_admin, "temp_esmr", esm, overwrite = TRUE)
DBI::dbExecute(con_diaspara_admin, "INSERT INTO refbast.tr_estimationmethod_esm
(esm_id, esm_code, esm_description, esm_icesvalue,  esm_icestablesource)
SELECT esm_id, esm_code, esm_description, esm_icesvalue, esm_icestablesource
FROM temp_esmr")# 8
DBI::dbExecute(con_diaspara_admin, "DROP table temp_esmr")
```

> **!** The full WGEEL database was imported in 2025. In 2026 there will be one last datacall using the previous database and shiny interface before switching to the new format.
> The remaining issues are here :
> Integrate the latest version of WGEEL database #29

6.3 Create datbast.t_metadata_met

Note there is a slight different here, I need to add twice the variables for salmon, and then for trutta. These are no duplicates as the primary key is set on both 127187 (Salmo trutta) and 127186 (Salmon salar). Later on, there should be variables only with 127186 (at the second step of the integration process.)

```
# t_metadata_met


df_all <- readxl::read_xlsx(file.path(datawd, "WGBAST_2024_Catch_29-02-2024.xlsx"), sheet =
df_all <- janitor::clean_names(df_all)
# C. (Cedric) From there following henni's script : https://github.com/hennip/WGBAST/blob/ma
# Comments with C. are added by Cedric, otherwise taken from github
```

```r
# quick fix to avoid logical I put char in subdiv_IC[1]
df_all$subdiv_ic[1] <- NA
df_all <- df_all |>
  mutate(tp_type=ifelse(tp_type=="QRT", "QTR", tp_type),
         gear=ifelse(gear=="GNS", "MIS", gear), # Tapani comment
         w_type = ifelse(w_type %in% c('EXP','GST'), 'EXV', w_type),
         n_type = ifelse(n_type %in% c('EXP','GST'), 'EXV', n_type))


# Gears

#unique(df_all$gear)
#NA    "AN"  "LLD" "GND" "MIS" "FYK" "All" "ALV"
#table(df_all$gear)
#All  ALV   AN  FYK  GND  LLD  MIS
#  93   29 1597 3432 1013 1541 9122

# driftnet=GND, longline=LLD, trapnet=FYK, angling=AN, other=MIS, set gillnet (anchored, sta

df_all$gearICES <- case_when(df_all$gear == "AN" ~ "LHP", # CHECK THIS Handlines and hand-op
                             df_all$gear == "LLD" ~ "LLD",
                             df_all$gear == "GND" ~ "GND",
                             df_all$gear == "MIS" ~ "MIS",
                             df_all$gear == "FYK" ~ "FYK",
                             df_all$gear == "ALV" ~ "LHP")   # LV and FI, in river RECR
# ALV: discarded alive, BMS: below minimum landing size (dead)


# creating t_metadata_met
#
# commercial=COMM, recreational=RECR, discard=DISC, sealdamage=SEAL, unreported=UNRP, ALV=re

#table(df_all$f_type, useNA = "ifany")
#  ALV   BMS BROOD  COMM  DISC  RECR  SEAL <NA>
#  537    77    39 12920   334  2473   980 368
#
#table(df_all$w_type)
# there are missing values for f_type, correspond to SAL FI/SE, 1972-1999
# then SAL 2000 FI/SE, 24-31 logbooks weights
# then SAL 2001 SE, 2000
# 3000 logbooks weights
# then 2 lines SAL TRS
# then lines for LT or LV
#
# => I think all those lines are COMM, this would be consistent with f_type in scripts
```

```
# where COMM is never used (the default)
#
#print(df_all[is.na(df_all$f_type), ], n ="Inf")

df_all <-
  bind_rows(
  df_all |>
  filter(is.na(f_type)) |>
  mutate(tp_type=ifelse(fishery == "F", "REC", "COMM")),
    df_all |>
  filter(!is.na(f_type)))


#  EST   EXP   EXT   EXV   GST   LOG
#  893    28   495  1218    30 14188
#
#  EST   EXP   EXT   EXV   GST   LOG
#  944    28   335  1295     9 14117

# table(df_all$n_type, df_all$w_type, useNA = "ifany")
  #       EST   EXT   EXV   LOG  <NA>
  # EST    679     2    26   181    56
  # EXP      0     0    28     0     0
  # EXT      2   256     0    75     2
  # EXV     51     0  1210     0    34
  # GST      1     0     8     0     0
  # LOG    106   237     0 13704    70
  # <NA>    54     0     4   228   714

# these are not used (f_type, w_type) => ignored or add to comments.
#
 table(df_all$f_type, df_all$w_type, useNA = "ifany")


#table(df_all$f_type)

saveRDS(df_all, file="data/wgbast_landings_modified.Rds")



# t_metadata_met  <-  dbGetQuery(con_diaspara, "SELECT * FROM datbast.t_metadata_met;")
# clipr::write_clip(colnames(t_metadata_met))
# head(t_metadata_met)
uktyp <- unique(df_all$f_type)
uktyp[is.na(uktyp)] <- "HIST" # historical data, check hopefully it's OK
```

```r
typ <- outer(c("N","W","E"), paste0("_",uktyp ), FUN = "paste0")
dim(typ) <- NULL
#"N_HIST"  "W_HIST"  "E_HIST"  "N_COMM"  "W_COMM"  "E_COMM"  "N_ALV"   "W_ALV"   "E_ALV"   "
#"W_DISC"  "E_DISC"  "N_SEAL"  "W_SEAL"  "E_SEAL"  "N_BMS"   "W_BMS"   "E_BMS"   "N_BROOD" "
#typ <- outer(typ, c("_SAL", "_TRS"),  FUN = "paste0")
#dim(typ) <- NULL


#get the type back
met_type <- substring(typ, 3,nchar(typ)) #COMM COMM, ....,

uni_code <- substring(typ, 1,1)
uni_code <- case_when(uni_code == "E" ~ "nd",
                      uni_code == "W" ~ "kg",
                      uni_code == "N" ~ "nr") #see https://diaspara.bordeaux-aquitaine.inrae



# vector with SAL ... then TRS
# The column species is repeated because the foreign key for t_stock_sto is on both TRS and
# so for instance we'll have two lines one with COMM_N for TRS one for SAL.
# It might seem weird but might allow for different metadata, and also most importantly
# later on, in the model some variables will be specific to SAL
# but when the variables are in common, then need to be repeated.
t_metadata_met_TRS  <- data.frame(
    met_var = typ,
    met_spe_code = "127187",
    met_wkg_code = "WGBAST",
    met_ver_code = "WGBAST-2025-1",
    met_oty_code = "Single_value", #  https://diaspara.bordeaux-aquitaine.inrae.fr/deliverab
    met_nim_code =  "Data", # https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/
    met_dim = paste0(
      "{", 1, ",",
       0, ",",
       0, "}"
    ),
    met_dimname = paste0(
      "{'NULL',NULL,NULL}"
    ),
    # Here unlike the eel, I cannot be sure the first dimension is year, might be MON, HYR
    met_modelstage = NA,
    met_type = met_type,
    # not a referential, used for legacy in WGNAS,
    # see https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p4/wgnas_salmoglob_c
    met_location = NA, # something line bef. Fisheries Aft fisheries.... not a referential
```

```
    met_fishery = NA, # not a referential
    met_mtr_code = NA, # reference to tr_metrictype (bound, mean, SD, can be left empty)
    met_des_code = case_when(
      met_type == "COMM" ~ "Removed",
      met_type == "ALV" ~ "Released",
      met_type == "RECR" ~ "Removed",
      met_type == "DISC" ~ "Discarded",
      met_type == "BROOD" ~ "Removed",
      met_type == "SEAL" ~ "Seal damaged",
      met_type == "BMS" ~ "Discarded",
      .default = NA),
      # https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p7stock/midb.html#dest
    met_uni_code = uni_code,
    met_cat_code = case_when(
       met_type == "COMM" ~ "Catch",
       met_type == "ALV" ~ "Release",
       met_type == "RECR" ~ "Catch",
       met_type == "DISC" ~ "Catch",
       met_type == "BROOD" ~ "Other",
       met_type == "SEAL" ~ "Other",
       met_type == "BMS" ~ "Catch",
       .default = NA),
met_definition = "TODO",
met_deprecated = NA
# not integrating any of the deprecated data
)
t_metadata_met_SAL <- t_metadata_met_TRS
t_metadata_met_SAL$met_spe_code<- "127186"


t_metadata_met <- bind_rows(t_metadata_met_TRS,t_metadata_met_SAL)

DBI::dbWriteTable(con_diaspara_admin, "temp_wgbast_t_metadata_met", t_metadata_met, overwrit


DBI::dbExecute(con_diaspara_admin, "DELETE FROM datbast.t_metadata_met")
DBI::dbExecute(con_diaspara_admin, "INSERT INTO datbast.t_metadata_met(met_var, met_spe_code
SELECT met_var, met_spe_code, met_wkg_code, met_ver_code, met_oty_code, met_nim_code, met_di
```

```
# sent by Becky
scalar <- read_csv("data/WGBAST scalars1.csv")
scalar <- janitor::clean_names(scalar)
array <- read_csv("data/WGBAST vectors_arrays1.csv")
array <- janitor::clean_names(array)
t_metadata_met_a  <- data.frame(
```

```
    met_var = array$variable_name,
    met_spe_code = "127186",
    met_wkg_code = "WGBAST",
    met_ver_code = "WGBAST-2025-1",
    met_oty_code = ifelse(is.na(array$dim2), "Vector",ifelse(is.na(array$dim3),"Matrix", "Ar
    met_nim_code =  ifelse(array$type == "stockastic", "Data", "Output"), #scenarios or stoc
    met_dim = paste0(
      "{", array$dim1, ",",
       ifelse(is.na(array$dim2), 0,array$dim2), ",",
       ifelse(is.na(array$dim3), 0,array$dim3), "}"
    ),
    met_dimname = paste0(
      "{",array$dim1_description,",", ifelse(array$dim2_description=="NA", NULL,array$dim2_d
    ),
    # Here unlike the eel, I cannot be sure the first dimension is year, might be MON, HYR .
    met_modelstage = NA,
    met_type = NA,  # TODO
    # not a referential, used for legacy in WGNAS,
    # see https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p4/wgnas_salmoglob_d
    met_location = NA, # something line bef. Fisheries Aft fisheries.... not a referential
    met_fishery = NA, # not a referential
    met_mtr_code = NA, # reference to tr_metrictype (bound, mean, SD, can be left empty)
    met_des_code = NA,
      # https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p7stock/midb.html#dest
    met_uni_code = NA,
    met_cat_code = NA, # TODO
met_definition = "TODO",
met_deprecated = NA
# not integrating any of the deprecated data
)
t_metadata_met_s  <- data.frame(
    met_var = scalar$name,
    met_spe_code = "127186",
    met_wkg_code = "WGBAST",
    met_ver_code = "WGBAST-2025-1",
    met_oty_code =  "Single_value",
    met_nim_code =  ifelse(scalar$type == "stochastic", "Data", "Output"), #scenarios or sto
    met_dim = paste0(
      "{", 1, ",",
       0, ",",
       0, "}"
    ),
    met_dimname = paste0(
      "{'NULL',NULL,NULL}"
    ),
```

```
    # Here unlike the eel, I cannot be sure the first dimension is year, might be MON, HYR
    met_modelstage = NA,
    met_type = NA,  # TODO
    # not a referential, used for legacy in WGNAS,
    # see https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p4/wgnas_salmoglob_d
    met_location = NA, # something line bef. Fisheries Aft fisheries.... not a referential
    met_fishery = NA, # not a referential
    met_mtr_code = NA, # reference to tr_metrictype (bound, mean, SD, can be left empty)
    met_des_code = NA,
      # https://diaspara.bordeaux-aquitaine.inrae.fr/deliverables/wp3/p7stock/midb.html#dest
    met_uni_code = NA,
    met_cat_code = NA, # TODO
met_definition = "TODO",
met_deprecated = NA
# not integrating any of the deprecated data
)
t_metadata_met<- bind_rows(t_metadata_met_s, t_metadata_met_a)
DBI::dbWriteTable(con_diaspara_admin, "temp_wgbast_t_metadata_met", t_metadata_met, overwrit


DBI::dbExecute(con_diaspara_admin, "INSERT INTO datbast.t_metadata_met(met_var, met_spe_code
SELECT met_var, met_spe_code, met_wkg_code, met_ver_code, met_oty_code, met_nim_code, met_di
```

6.4 datbast.t_stock_sto

There are three additional column in `databast.t_stock_sto` when compared to `dat.t_stock_sto`, the table from which it inherits. This is similar to `datnas.t_stock_sto` where an additional column was created to handle the extra dimension for some arrays stored in WGNAS. The columns are :

- sto_tip_code the time period, one of YR, HYR (half year), QTR (Quarter), MON (Month). A vocabulary has been created for checks on these time periods.
- sto_timeperiod integer, the value of the time period. Note : a trigger has been created to handle different possible values for sto_tip_code (e.g. half of year can be 1 or 2 , and month between 1 and 12).
- sto_datasourcecode. This column is not used in scripts, but discussion with Henni have shown that this remains important. It will be adapted to ICES vocab `DataSource` with additions for elements on the calculation of smolts in the Young fish database (see

1) .

SQL code to create table `datbast.t_stock_sto`

```
-- CREATE A TABLE INHERITED FROM dat.t_stock_sto.
-- Table dat.stock_sto only gets data by inheritance.
-- Here we have to build the constraints again.
-- delete from datbast.t_stock_sto;
DROP TABLE IF EXISTS datbast.t_stock_sto;
CREATE TABLE datbast.t_stock_sto (
    sto_gear_code text,
  CONSTRAINT fk_sto_gear_code FOREIGN KEY (sto_gear_code)
    REFERENCES  ref.tr_gear_gea(gea_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    sto_tip_code TEXT,
  CONSTRAINT fk_tip_code FOREIGN KEY (sto_tip_code)
    REFERENCES  ref.tr_timeperiod_tip(tip_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
    sto_timeperiod integer NOT NULL,
    sto_dts_code TEXT,
  CONSTRAINT fk_sto_dts_code FOREIGN KEY (sto_dts_code)
    REFERENCES ref.tr_datasource_dts(dts_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
   sto_dtb_code TEXT,
  CONSTRAINT fk_sto_dtb_code FOREIGN KEY (sto_dtb_code)
    REFERENCES ref.tr_databasis_dtb(dtb_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  sto_esm_code TEXT,
  CONSTRAINT fk_sto_esm_code FOREIGN KEY (sto_esm_code)
    REFERENCES refbast.tr_estimationmethod_esm(esm_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_met_var_met_spe_code
    FOREIGN KEY (sto_met_var, sto_spe_code)
    REFERENCES datbast.t_metadata_met(met_var,met_spe_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_are_code FOREIGN KEY (sto_are_code)
    REFERENCES refbast.tr_area_are (are_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_cou_code FOREIGN KEY (sto_cou_code)
    REFERENCES ref.tr_country_cou (cou_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_lfs_code_sto_spe_code FOREIGN KEY (sto_lfs_code, sto_spe_code)
    REFERENCES ref.tr_lifestage_lfs (lfs_code, lfs_spe_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_hty_code FOREIGN KEY (sto_hty_code)
    REFERENCES ref.tr_habitattype_hty(hty_code)
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_qal_code FOREIGN KEY (sto_qal_code)
    REFERENCES ref.tr_quality_qal(qal_code)
```

```
    ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_mis_code FOREIGN KEY (sto_mis_code)
  REFERENCES ref.tr_missvalueqal_mis (mis_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_dta_code FOREIGN KEY (sto_dta_code)
  REFERENCES ref.tr_dataaccess_dta(dta_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT fk_sto_wkg_code  FOREIGN KEY (sto_wkg_code)
  REFERENCES ref.tr_icworkinggroup_wkg(wkg_code)
  ON UPDATE CASCADE ON DELETE RESTRICT,
  CONSTRAINT c_uk_sto_id_sto_wkg_code UNIQUE (sto_id, sto_wkg_code),
  CONSTRAINT ck_notnull_value_and_mis_code CHECK ((((sto_mis_code IS NULL) AND (sto_value IS
  ((sto_mis_code IS NOT NULL) AND (sto_value IS NULL))))

)
inherits (dat.t_stock_sto) ;

-- This table will always be for WGBAST

ALTER TABLE datbast.t_stock_sto ALTER COLUMN sto_spe_code SET DEFAULT NULL;
ALTER TABLE datbast.t_stock_sto ADD CONSTRAINT ck_spe_code CHECK (sto_spe_code='127186' OR s
ALTER TABLE datbast.t_stock_sto ALTER COLUMN sto_wkg_code SET DEFAULT 'WGBAST';
ALTER TABLE datbast.t_stock_sto ADD CONSTRAINT ck_wkg_code CHECK (sto_wkg_code='WGBAST');



ALTER TABLE datbast.t_stock_sto OWNER TO diaspara_admin;
GRANT ALL ON TABLE datbast.t_stock_sto TO diaspara_read;




COMMENT ON TABLE datbast.t_stock_sto IS
'Table including the stock data in schema datbast.... This table feeds the dat.t_stock_sto t
to the catch excel table in the original WGBAST database.';
COMMENT ON COLUMN datbast.t_stock_sto.sto_id IS 'Integer serial identifying. Only unique in
when looking at the pair, sto_id, sto_wkg_code';
COMMENT ON COLUMN datbast.t_stock_sto.sto_gear_code IS 'Code of the gear used, this column i
COMMENT ON COLUMN datbast.t_stock_sto.sto_tip_code IS 'Code of the time period used, one of
COMMENT ON COLUMN datbast.t_stock_sto.sto_timeperiod IS 'An integer giving the value of the
COMMENT ON COLUMN datbast.t_stock_sto.sto_dts_code IS 'Code of the data source one of Logb (
COMMENT ON COLUMN datbast.t_stock_sto.sto_dtb_code IS 'Code of the data basis, one of Estima
COMMENT ON COLUMN datbast.t_stock_sto.sto_esm_code IS 'Code of the estimation method, one of
COMMENT ON COLUMN datbast.t_stock_sto.sto_met_var IS 'Name of the variable in the database,
on the pair of column sto_spe_code, sto_met_var';
```

```
-- note if we end up with a single table, then the constraint will  have to be set
-- on sto_wkg_code, sto_spe_code and sto_met_code.
COMMENT ON COLUMN datbast.t_stock_sto.sto_year IS 'Year';
COMMENT ON COLUMN datbast.t_stock_sto.sto_value IS 'Value if null then provide a value in st
COMMENT ON COLUMN datbast.t_stock_sto.sto_are_code IS 'Code of the area, areas are geographi
see tr_area_are.';
COMMENT ON COLUMN datbast.t_stock_sto.sto_cou_code IS 'Code of the country see tr_country_co
COMMENT ON COLUMN datbast.t_stock_sto.sto_lfs_code IS 'Code of the lifestage see tr_lifestag
both lfs_code, and lfs_spe_code (as two species can have the same lifestage code.';
COMMENT ON COLUMN datbast.t_stock_sto.sto_hty_code IS 'Code of the habitat type, one of MO (
T (Transitional water), FW (Freshwater), null accepted';
COMMENT ON COLUMN datbast.t_stock_sto.sto_qal_code IS 'Code of data quality (1 good quality,
3 bad quality (not used), 4 dubious, 18, 19 ... historical data not used.
Not null, Foreign key set to tr_quality_qal';
COMMENT ON COLUMN datbast.t_stock_sto.sto_qal_comment IS 'Comment for the quality, for insta
COMMENT ON COLUMN datbast.t_stock_sto.sto_comment IS 'Comment on the value';
COMMENT ON COLUMN datbast.t_stock_sto.sto_datelastupdate IS 'Last update of the data';
COMMENT ON COLUMN datbast.t_stock_sto.sto_mis_code IS 'When no value are given in sto_value,
references table tr_missvalueqal_mis, should be null if value is provided (can''t have both)
COMMENT ON COLUMN datbast.t_stock_sto.sto_dta_code IS 'Access to data, default is ''Public''
COMMENT ON COLUMN datbast.t_stock_sto.sto_wkg_code IS 'Code of the working group, one of
WGBAST, WGEEL, WGNAS, WKTRUTTA';
COMMENT ON COLUMN datbast.t_stock_sto.sto_ver_code IS 'Version code, references refbast.tr_v

-- trigger on date
DROP FUNCTION IF EXISTS datbast.update_sto_datelastupdate CASCADE;
CREATE OR REPLACE FUNCTION datbast.update_sto_datelastupdate()
 RETURNS trigger
 LANGUAGE plpgsql
AS $function$
BEGIN
    NEW.sto_datelastupdate = now()::date;
    RETURN NEW;
END;
$function$
;

ALTER FUNCTION datbast.update_sto_datelastupdate() OWNER TO diaspara_admin;


CREATE TRIGGER update_sto_datelastupdate BEFORE
INSERT
    OR
UPDATE
    ON
```

```
    datbast.t_stock_sto FOR EACH ROW EXECUTE FUNCTION datbast.update_sto_datelastupdate();



-- trigger to check consistency between sto_timeperiod and sto_tip_code
-- Mon
CREATE OR REPLACE FUNCTION datbast.check_time_period()
 RETURNS TRIGGER
 LANGUAGE plpgsql
AS $check_time_period$
BEGIN
   -- sto_timeperiod is always positive or NULL
   -- if sto_tip_code = "YR" keep sto_timeperiod NULL
    IF (NEW.sto_tip_code = 'Year' AND NEW.sto_timeperiod > 0) THEN
    RAISE EXCEPTION 'sto_timeperiod should be 0 when the code for time period (sto_tip_code)
    END IF;
     -- if sto_tip_code = "QTR" check 1 2 3 or 4
    IF (NEW.sto_tip_code = '>Quarter' AND NEW.sto_timeperiod > 4) THEN
    RAISE EXCEPTION 'sto_timeperiod should be 1, 2, 3 or 4 for quarters';
    END IF;
    -- half of year is one or two
    IF (NEW.sto_tip_code = 'Half of Year' AND NEW.sto_timeperiod > 2) THEN
    RAISE EXCEPTION 'sto_timeperiod should be 1 (first half) 2 (second half) when the code f
    END IF;
    -- half of year is one or two
    IF (NEW.sto_tip_code = 'Month' AND NEW.sto_timeperiod > 12) THEN
    RAISE EXCEPTION 'sto_timeperiod should be 1 to 12 when the code for timeperiod (sto_tip_
    END IF;
    RETURN NEW;
END;
$check_time_period$;

ALTER FUNCTION datbast.check_time_period() OWNER TO diaspara_admin;

DROP TRIGGER IF EXISTS trg_check_time_period ON datbast.t_stock_sto ;
CREATE TRIGGER trg_check_time_period BEFORE
INSERT
    OR
UPDATE
    ON
    datbast.t_stock_sto FOR EACH ROW EXECUTE FUNCTION datbast.check_time_period();
```

Clearly the table of datasource will have to be revised and new methodologies

added and

```
df_all <- readRDS(file="data/wgbast_landings_modified.Rds")


# Modify some lines with comments (TO BE CHECKED BY WGBAST)
df_all[df_all$sub_div == "21" & ! is.na(df_all$sub_div) ,"Notes"] <- "ATTENTION THESE WERE N

#Inversion of tp_type and n_type
df_all[df_all$tp_type=='COMM'& df_all$time_period!=0,"tp_type"] <- "MON" # 4 lines where CON
# the other are year
id_err <- which(is.na(df_all$f_type)& df_all$tp_type=="COMM")
df_all[id_err,"tp_type"] <- "YR"
df_all[id_err,"f_type"] <- "COMM"
#Year without 0 as time_period one line with 2 in Estonia all other have 0
df_all[df_all$tp_type=='YR'& df_all$time_period!=0,"time_period"]<- 0
# Year should be YR in the initial dataset
df_all[df_all$tp_type=="Year" & !is.na(df_all$tp_type),"tp_type"] <- "YR"
# missing tp_type
df_all[is.na(df_all$tp_type),"tp_type"] <- "YR" # 6 lines with historical data


# in Estonia there are both SAL&TRS and NA, which correspond to sea damage unspecified.
# I cannot have that, will report as Salmon, leave to the WGBAST to see how to handle this.
df_all[!is.na(df_all$species) & df_all$species =="SAL&TRS","species"]<- "127186"
df_all[is.na(df_all$species) ,"species"]<- "127186"
df_all[df_all$species=="NA" ,"species"]<- "127186"




# create table of rivernames in WGBAST, do queries and manual search to join them
# to our layer.
# tt <- table(df_all$river)
# tt <- tt[order(tt, decreasing =TRUE)]
# tt <- as.data.frame(tt)
# colnames(tt) <- c("riv_are_name", "number")
# dbWriteTable(con_diaspara_admin, "landings_wbast_river_names", tt,overwrite = TRUE)
# dbExecute(con_diaspara_admin, "ALTER TABLE landings_wbast_river_names set schema refbast;"
# dbExecute(con_diaspara_admin, "ALTER TABLE refbast.landings_wbast_river_names ADD COLUMN r
#
# For recreational fisheries, the river column is used.
# It will be used as the hierarchy level to enter the data
riv <- dbGetQuery(con_diaspara_admin, "SELECT * FROM refbast.landings_wbast_river_names
                  JOIN refbast.tr_area_are on are_code = riv_are_code")
```

```r
# get the are_code...
df_all <- df_all |>
  left_join(riv |> select(riv_are_name, riv_are_code), by = join_by(river == riv_are_name))

gear <- dbGetQuery(con_diaspara_admin, "SELECT * FROM ref.tr_gear_gea WHERE gea_icesvalue is

df_all <- df_all |>
  left_join(gear |> select(gea_code, gea_icesvalue), by = join_by(gearICES == gea_icesvalue)


# Insert Numbers
df_all_N <- df_all |>
  filter(!is.na(numb)) |>
  mutate(f_type = ifelse(is.na(f_type), "HIST", f_type)) |>
  mutate(sto_met_var = paste0("N_",f_type))

# unit allows to avoid having NA in strings ... Here we put additional info in the comment f
df_all_N$n_type2 <- df_all_N$n_type
df_all_N$n_type2[!is.na(df_all_N$n_type)]<-paste0("N_type=",df_all_N$n_type2[!is.na(df_all_N
df_all_N$n_type2[!is.na(df_all_N$notes)] <-paste0(", ",df_all_N$n_type2[!is.na(df_all_N$note
df_all_N$n_type2[is.na(df_all_N$n_type)]<- ""
df_all_N$notes2 <- df_all_N$notes
df_all_N$notes2[is.na(df_all_N$notes2)] <- ""
df_all_N$notes2 <- paste0(df_all_N$notes2,df_all_N$n_type2)


t_stock_sto_N = data.frame(
  sto_met_var = df_all_N$sto_met_var,
  sto_year = df_all_N$year,
  sto_spe_code = df_all_N$species,
  sto_value = df_all_N$numb,
  # if it's a river get the code of the river otherwise get other codes....
  sto_are_code = case_when(!is.na(df_all_N$river) ~ df_all_N$riv_are_code,
                           df_all_N$sub_div == "22-32" ~ "27.3.d", # correct, 3 lines corres
                           df_all_N$sub_div == "200" ~ "27.3.d.22-29",
                           df_all_N$sub_div == "300" ~ "27.3.d.30-31",
                           df_all_N$sub_div == "32"  ~  "27.3.d.32",
                           df_all_N$sub_div == "31"  ~  "27.3.d.31",
                           df_all_N$sub_div == "30"  ~  "27.3.d.30",
                           df_all_N$sub_div == "29"  ~  "27.3.d.29",
                           df_all_N$sub_div == "27"  ~  "27.3.d.27",
                           df_all_N$sub_div == "26"  ~  "27.3.d.26",
                           df_all_N$sub_div == "25"  ~  "27.3.d.25",
                           df_all_N$sub_div == "24"  ~  "27.3.d.24",
                           df_all_N$sub_div == "23"  ~  "27.3.b.23",
```

```
                          df_all_N$sub_div == "22"  ~  "27.3.c.22",
                          df_all_N$sub_div == "21"  ~  "27.3.d.31",# 3 Lines for sweden con
                          # here we allow for the two subdivision in the Baltic
                          df_all_N$sub_div == "28" & df_all_N$subdiv_ic == '27.3.d.28.1' ~
                          df_all_N$sub_div == "28" & df_all_N$subdiv_ic == '27.3.d.28.2' ~
                          df_all_N$sub_div == "28"  ~  "27.3.d.28",
                          is.na(df_all_N$sub_div)   ~   "27.3.d" # 12 rows with with commen
                          ),
sto_cou_code = df_all_N$country,
sto_lfs_code = 'A',
sto_hty_code = case_when(df_all_N$fishery == "O" ~ "MO",
                          df_all_N$fishery == "C"~  "MC",
                          df_all_N$fishery == "R" ~ "FW"),
sto_qal_code = 1,
sto_comment = df_all_N$notes2,
sto_datelastupdate = Sys.Date(),
sto_mis_code = NA,
sto_dta_code = "Public", # check this
sto_wkg_code = "WGBAST",
sto_ver_code = "WGBAST-2025-1",
sto_gear_code  = df_all_N$gea_code,
sto_tip_code  = case_when(df_all_N$tp_type == "YR" ~"Year",
                          df_all_N$tp_type == "HYR" ~ "Half of Year",
                          df_all_N$tp_type == "MON" ~ "Month",
                          df_all_N$tp_type == "MONTH" ~ "Month",
                          df_all_N$tp_type == "QTR" ~ "Quarter",
                          df_all_N$tp_type == "COMM" ~ "Quarter", # this is an error
                          is.na( df_all_N$tp_type) ~ "Year",
                          .default = "Troube this will fail at insertion"),
sto_timeperiod = df_all_N$time_period,

# in the notes some elements hint at surveys, but this will need a check up by WGBAST anyw
sto_dts_code = case_when(df_all_N$n_type == "LOG" ~ "Logb",
                                df_all_N$n_type == "EXV" ~ "Exprt",
                                df_all_N$n_type == "EST" & (grepl("survey",tolower(df_all_N
                                                            grepl("question",tolower(df_all_N
                                                            grepl("query", tolower(df_all_N$n
                                                            grepl("web", tolower(df_all_N$not
                               df_all_N$n_type == "EXT" ~ NA),
sto_dtb_code = case_when(df_all_N$n_type == "LOG" ~ "Official",
                          df_all_N$n_type == "EXV"  ~  "Estimated",
                          df_all_N$n_type == "EST" ~ "Estimated",
                          df_all_N$n_type == "EXT" ~ "Estimated",
                          .default =  "Unknown"),
sto_esm_code = NA
```

```
)


#dbExecute(con_diaspara_admin, "drop table if exists temp_t_stock_sto_n")
dbExecute(con_diaspara_local, "drop table if exists temp_t_stock_sto_n")
system.time(dbWriteTable(con_diaspara_local, "temp_t_stock_sto_n", t_stock_sto_N)) # 27s
dbExecute(con_diaspara_local, "DELETE FROM datbast.t_stock_sto")
dbExecute(con_diaspara_local, "INSERT INTO datbast.t_stock_sto(sto_met_var, sto_year, sto_sp
          SELECT sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code, sto_cou_code,
dbExecute(con_diaspara_local, "drop table temp_t_stock_sto_n")


# Insert WEIGHTS-----------------------------------
#
df_all_W <- df_all |>
  filter(!is.na(weight)) |>
  mutate(f_type = ifelse(is.na(f_type), "HIST", f_type)) |>
  mutate(sto_met_var = paste0("W_",f_type))

# unit allows to avoid having NA in strings ... Here we put additional info in the comment f
df_all_W$w_type2 <- df_all_W$w_type
df_all_W$w_type2[!is.na(df_all_W$w_type)]<-paste0("W_type=",df_all_W$w_type2[!is.na(df_all_W
df_all_W$w_type2[!is.na(df_all_W$notes)] <- paste0(", ",df_all_W$w_type2[!is.na(df_all_W$not
df_all_W$w_type2[is.na(df_all_W$w_type)]<- ""
df_all_W$notes2 <- df_all_W$notes
df_all_W$notes2[is.na(df_all_W$notes2)] <- ""
df_all_W$notes2<- paste0(df_all_W$notes2,df_all_W$w_type2)
df_all_W$notes2[df_all_W$notes2==""] <- NA

t_stock_sto_W = data.frame(
  sto_met_var = df_all_W$sto_met_var,
  sto_year = df_all_W$year,
  sto_spe_code = df_all_W$species,
  sto_value = df_all_W$weight,
  # if it's a river get the code of the river otherwise get other codes....
  sto_are_code = case_when(!is.na(df_all_W$river) ~ df_all_W$riv_are_code,
                           df_all_W$sub_div == "22-32" ~ "27.3.d", # correct, 3 lines corres
                           df_all_W$sub_div == "200" ~ "27.3.d.22-29",
                           df_all_W$sub_div == "300" ~ "27.3.d.30-31",
                           df_all_W$sub_div == "32"  ~  "27.3.d.32",
                           df_all_W$sub_div == "31"  ~  "27.3.d.31",
                           df_all_W$sub_div == "30"  ~  "27.3.d.30",
                           df_all_W$sub_div == "29"  ~  "27.3.d.29",
                           df_all_W$sub_div == "27"  ~  "27.3.d.27",
                           df_all_W$sub_div == "26"  ~  "27.3.d.26",
```

```
                          df_all_W$sub_div == "25"  ~  "27.3.d.25",
                          df_all_W$sub_div == "24"  ~  "27.3.d.24",
                          df_all_W$sub_div == "23"  ~  "27.3.b.23",
                          df_all_W$sub_div == "22"  ~  "27.3.c.22",
                          df_all_W$sub_div == "21"  ~  "27.3.d.31",# 3 Lines for sweden con
                          # here we allow for the two subdivision in the Baltic
                          df_all_W$sub_div == "28" & df_all_W$subdiv_ic == '27.3.d.28.1' ~
                          df_all_W$sub_div == "28" & df_all_W$subdiv_ic == '27.3.d.28.2' ~
                          df_all_W$sub_div == "28"  ~  "27.3.d.28",
                          is.na(df_all_W$sub_div)   ~   "27.3.d" # 12 rows with with commer
                          ),
sto_cou_code = df_all_W$country,
sto_lfs_code = 'A',
sto_hty_code = case_when(df_all_W$fishery == "O" ~ "MO",
                          df_all_W$fishery == "C"~  "MC",
                          df_all_W$fishery == "R" ~ "FW"),
sto_qal_code = 1,
sto_comment = df_all_W$notes2,
sto_datelastupdate = Sys.Date(),
sto_mis_code = NA,
sto_dta_code = "Public", # check this
sto_wkg_code = "WGBAST",
sto_ver_code = "WGBAST-2025-1",
sto_gear_code  = df_all_W$gea_code,
sto_tip_code  = case_when(df_all_W$tp_type == "YR" ~"Year",
                           df_all_W$tp_type == "HYR" ~ "Half of Year",
                           df_all_W$tp_type == "MON" ~ "Month",
                           df_all_W$tp_type == "MONTH" ~ "Month",
                           df_all_W$tp_type == "QTR" ~ "Quarter",
                           df_all_W$tp_type == "COMM" ~ "Quarter", # this is an error
                           is.na( df_all_W$tp_type) ~ "Year",
                           .default = "Troube this will fail at insertion"),
sto_timeperiod = df_all_W$time_period,

# in the notes some elements hint at surveys, but this will need a check up by WGBAST anyw
sto_dts_code = case_when(df_all_W$n_type == "LOG" ~ "Logb",
                              df_all_W$n_type == "EXV" ~ "Exprt",
                              df_all_W$n_type == "EST" & (grepl("survey",tolower(df_all_W
                                                          grepl("question",tolower(df_all_W
                                                          grepl("query", tolower(df_all_W$n
                                                          grepl("web", tolower(df_all_W$not
                             df_all_W$n_type == "EXT" ~ NA),
sto_dtb_code = case_when(df_all_W$n_type == "LOG" ~ "Official",
                          df_all_W$n_type == "EXV"  ~  "Estimated",
                          df_all_W$n_type == "EST" ~ "Estimated",
```

```
                            df_all_W$n_type == "EXT" ~ "Estimated",
                            .default =  "Unknown"),
  sto_esm_code = NA
)

#dbExecute(con_diaspara_admin, "drop table if exists temp_t_stock_sto_w")
dbExecute(con_diaspara_local, "drop table if exists temp_t_stock_sto_w")
system.time(dbWriteTable(con_diaspara_local, "temp_t_stock_sto_w", t_stock_sto_W)) # 0.14
dbExecute(con_diaspara_local, "INSERT INTO datbast.t_stock_sto(sto_met_var, sto_year, sto_sp
          SELECT sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code, sto_cou_code,
dbExecute(con_diaspara_local, "drop table temp_t_stock_sto_w")

# Insert Effort


df_all_e <- df_all |>
  filter(!is.na(effort)) |>
  mutate(f_type = ifelse(is.na(f_type), "HIST", f_type)) |>
  mutate(sto_met_var = paste0("E_",f_type))

# unit allows to avoid having NA in strings ... Here we put additional info in the comment f
df_all_e$w_type2 <- df_all_e$w_type
df_all_e$w_type2[!is.na(df_all_e$w_type)]<-paste0("W_type=",df_all_e$w_type2[!is.na(df_all_e
df_all_e$w_type2[!is.na(df_all_e$notes)] <-paste0(", ",df_all_e$w_type2[!is.na(df_all_e$note
df_all_e$w_type2[is.na(df_all_e$w_type)]<- ""
df_all_e$notes2 <- df_all_e$notes
df_all_e$notes2[is.na(df_all_e$notes2)] <- ""
df_all_e$notes2<- paste0(df_all_e$notes2,df_all_e$w_type2)
df_all_e$notes2[df_all_e$notes2==""] <- NA

t_stock_sto_e = data.frame(
  sto_met_var = df_all_e$sto_met_var,
  sto_year = df_all_e$year,
  sto_spe_code = df_all_e$species,
  sto_value = df_all_e$effort,
  # if it's a river get the code of the river otherwise get other codes....
  sto_are_code = case_when(!is.na(df_all_e$river) ~ df_all_e$riv_are_code,
                            df_all_e$sub_div == "22-32" ~ "27.3.d", # correct, 3 lines corres
                            df_all_e$sub_div == "200" ~ "27.3.d.22-29",
                            df_all_e$sub_div == "300" ~ "27.3.d.30-31",
                            df_all_e$sub_div == "32"  ~  "27.3.d.32",
                            df_all_e$sub_div == "31"  ~  "27.3.d.31",
                            df_all_e$sub_div == "30"  ~  "27.3.d.30",
                            df_all_e$sub_div == "29"  ~  "27.3.d.29",
                            df_all_e$sub_div == "27"  ~  "27.3.d.27",
```

```
                          df_all_e$sub_div == "26"  ~  "27.3.d.26",
                          df_all_e$sub_div == "25"  ~  "27.3.d.25",
                          df_all_e$sub_div == "24"  ~  "27.3.d.24",
                          df_all_e$sub_div == "23"  ~  "27.3.b.23",
                          df_all_e$sub_div == "22"  ~  "27.3.c.22",
                          df_all_e$sub_div == "21"  ~  "27.3.d.31",# 3 Lines for sweden con
                          # here we allow for the two subdivision in the Baltic
                          df_all_e$sub_div == "28" & df_all_e$subdiv_ic == '27.3.d.28.1' ~
                          df_all_e$sub_div == "28" & df_all_e$subdiv_ic == '27.3.d.28.2' ~
                          df_all_e$sub_div == "28"  ~  "27.3.d.28",
                          is.na(df_all_e$sub_div)   ~   "27.3.d" # 12 rows with with commer
                          ),
  sto_cou_code = df_all_e$country,
  sto_lfs_code = 'A',
  sto_hty_code = case_when(df_all_e$fishery == "O" ~ "MO",
                          df_all_e$fishery == "C"~  "MC",
                          df_all_e$fishery == "R" ~ "FW"),
  sto_qal_code = 1,
  sto_comment = df_all_e$notes2,
  sto_datelastupdate = Sys.Date(),
  sto_mis_code = NA,
  sto_dta_code = "Public", # check this
  sto_wkg_code = "WGBAST",
  sto_ver_code = "WGBAST-2025-1",
  sto_gear_code  = df_all_e$gea_code,
  sto_tip_code  = case_when(df_all_e$tp_type == "YR" ~"Year",
                           df_all_e$tp_type == "HYR" ~ "Half of Year",
                           df_all_e$tp_type == "MON" ~ "Month",
                           df_all_e$tp_type == "MONTH" ~ "Month",
                           df_all_e$tp_type == "QTR" ~ "Quarter",
                           df_all_e$tp_type == "COMM" ~ "Quarter", # this is an error
                           is.na( df_all_e$tp_type) ~ "Year",
                           .default = "Troube this will fail at insertion"),
  sto_timeperiod = df_all_e$time_period,

  # in the notes some elements hint at surveys, but this will need a check up by WGBAST anyw
  sto_dts_code = case_when(df_all_e$n_type == "LOG" ~ "Logb",
                               df_all_e$n_type == "EXV" ~ "Exprt",
                               df_all_e$n_type == "EST" & (grepl("survey",tolower(df_all_e
                                                          grepl("question",tolower(df_all_e
                                                          grepl("query", tolower(df_all_e$n
                                                          grepl("web", tolower(df_all_e$not
                              df_all_e$n_type == "EXT" ~ NA),
  sto_dtb_code = case_when(df_all_e$n_type == "LOG" ~ "Official",
                          df_all_e$n_type == "EXV"  ~  "Estimated",
```

```
                            df_all_e$n_type == "EST" ~ "Estimated",
                            df_all_e$n_type == "EXT" ~ "Estimated",
                            .default =  "Unknown"),
  sto_esm_code = NA
)

#dbExecute(con_diaspara_admin, "drop table if exists temp_t_stock_sto_e")
dbExecute(con_diaspara_local, "drop table if exists temp_t_stock_sto_e")
system.time(dbWriteTable(con_diaspara_local, "temp_t_stock_sto_e", t_stock_sto_e)) # 0.14
dbExecute(con_diaspara_local, "INSERT INTO datbast.t_stock_sto(sto_met_var, sto_year, sto_sp
          SELECT sto_met_var, sto_year, sto_spe_code, sto_value, sto_are_code, sto_cou_code,
dbExecute(con_diaspara_local, "drop table temp_t_stock_sto_e")


# Insert N_CI

# This is a bit too difficult, it's not always consistent. Could do when values are sepearat
# but it's not always the case. There aren't that many values, should be checked.

# Insert W_CI
```

⚠ WGBAST corrections made to the dataset

3 lines with area 21 (which do not exists) => It's in sweden for TRS is it correct to assign it to 31 ? To be checked during integration.
tp_type has 89 lines with COMM (it should be a time period) => Assigned to Year, please check

QUESTION WGBAST: Would the data access be restricted ?

Data access can be Resticted or Public.

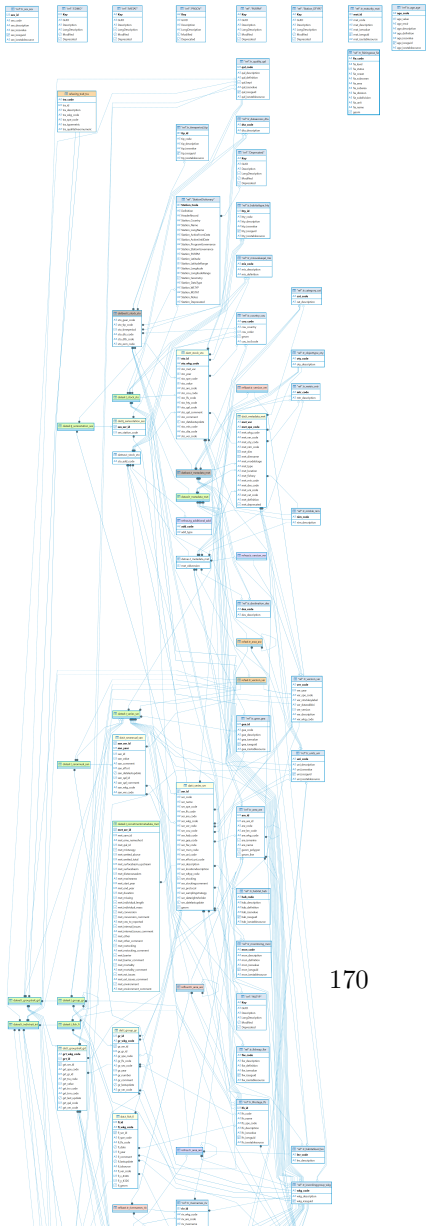
TODO WGBAST : n_type and w_type

Currently, we cannot easily translate all values “LOG” “EST” “EXV” “EXT” values with ICES vocab. While Logbook is OK. We think that you will need to create a dictionary of possible estimation methods (we could have more), and then resubmit your data while screening for the correct type. For instance, we don’t have an equivalent for EXT (extrapolated) so has not been translated, but the f_type and w_type are provided in the column comment. For “Est” when notes indicated “survey”, we have assessed it as SampDS. If you look at the tr_datasource_dts you will see a table of values in the ICES vocab. Among possible candidates are OthDF Other declarative forms (i.e. landing declarations and national declarative forms), or SampDC Commercial sampling data (sampling methodologies specific to each country). This refers to sampling in commercial vessels, not only for commercial species, or SampDS Survey sampling data (sampling methodologies specific to each country).

Chapter 7

Final diagram

Full diagram of the diadromous DB including both stock and metric



Chapter 8

Conclusions

A database was created for WGNAS, WGBAST, and WGEEL. It could easily have a similar format for WGTRUTTA and other working groups working with diadromous fish species, such as lampreys and shads. It will allow for the integration of data call files using DATSU. The habitat database is currently ported to RDBES and it has been proposed in RDBFIS to allow for the inclusion of data in the continental part of the range for migratory fishes.

Starting with the creation of formats for metrics in the summer 2025, this database will progressively be handed over to ICES. Since it was programmed in Postgres, this will require some work. In the long run, however, the use of a common database and ICES tools will: 1) simplify the learning needs of experts and knowledge, when shifting between different diadromous expert and assessment groups, 2) allow to provide a more streamlined data flow between national and international level, and 3) enhance accessibility and interoperability.

The shift to the new database will start in 2026 for WGEEL (metric DB) and probably 2028 for WGNAS and WGBAST depending on ressources to make the change. Changing from one database to another system will break the R code, which will need to be reviewed

! From now on everywhere in the db: Atlantic salmon will be 127186 and eel 126281. Sorry about that. Please bookmark the link to this paragraph in your browser to find it again.

spe_code	spe_commonname	spe_scientificname
127186	Atlantic salmon	Salmo salar
126413	Twait shad	Alosa alosa
126415	Allis shad	Alosa fallax
101174	Sea lamprey	Petromyzon marinus
101172	European river lamprey	Lampetra fluviatilis
126281	European eel	Anguilla anguilla
127187	Sea trout	Salmo trutta

Chapter 9

Acknowledgements and references

- Data source : EuroGeographics and UN-FAO for countries

ICES. 2024a. "The Second ICES/NASCO Workshop on Salmon Mortality at Sea (WKSalmon2; Outputs from 2022 Meeting)." ICES Scientific Reports. https://doi.org/10.17895/ICES.PUB.22560790.

———. 2024b. "Working Group on Biological Parameters (WGBIOP, Outputs from 2023 Meeting)," January. https://doi.org/10.17895/ices.pub.24961410.v2.