# INDEX

Notes:

On import check CODE exists in tbl\_taxonomic\_order

AND genus & species exists in respective combination

(Species can be moved in Bugs - between genera etc)

KEEP LOG OF ALL CHANGES IN BUGSCEP, BUT DO NOT TRACK THESE IN SEAD STRUCTURE

I.e. If a species changes CODE in BugsCEP, change the code in SEAD, and write a note in THE LOG (which will be put online eventually).

# TSeasonActiveAdult

Mapping

Code -> tbl\_taxonomic\_order.code => tbl\_taxa\_seasonality.taxon\_id

HSeason -> convert via text lookup from tbl\_seasons => tbl\_taxa\_seasonality.season\_id

CountryCode -> convert via text lookup from tbl\_locations.location\_name => tbl\_taxa\_seasonality.location\_id

lookup activity type from tbl\_activity\_types on ‘Adult active’ => tbl\_taxa\_seasonality.activity\_type\_id

requires

species + taxonomic order

countries

season notes

Se => ?

Sep => ?

How to deal with changes in datasets?

Currently only adding data, but the dataset for a species may change seasonal envelope. How to deal with the updated information: just add new data, or also remove the old?

This also applies to the use case of updating data that has been updated (what counts as being the updated information?, species, location + species?)

# TCountry

Mapping

Country -> tbl\_locations.location\_name

lookup tbl\_location\_types.location\_type\_id on location\_type= ‘Country’ => tbl\_locations.location\_type\_id

# TEcoKoch

Mapping

CODE => tbl\_taxonomic\_orders.code => tbl\_ecocodes.taxon\_id

BugsKochCode  => lookup tbl\_ecocode\_definitions.abbreviation => tbl\_ecocodes.ecocode\_definition\_id

requires

kochecodefinitions

# TEcoDefKoch

Mapping

BugsKochCode => tbl\_ecocode\_definitions.abbreviation

FullName => tbl\_ecocode\_definitions.label

Definition => tbl\_ecocode\_definitions.definition (YES)

Notes => tbl\_ecocode\_definitions.notes (YES)

[lookup tbl\_ecocode\_groups.label = ‘Koch group’ and system = ‘Kock system’ => tbl\_ecocode\_definitions.ecocode\_group\_id ??? not dependent on KochGroup]

tbl\_ecocode\_group\_id = tbl.ecocode\_groups.ecocode\_group\_id where TEcoDefKoch.KochGroup = TEcoDefGroups.EcoGroupCode (…)

# TEcoDefGroups

May not have thought this one through properly as we lack the abbreviation column in SEAD.

One solution is:

tbl\_ecocode\_groups.label = EcoName & " (" & EcoGroupCode & “)”

map this:

econame => tbl\_ecocode\_groups.label

ecogroupcode = > tbl\_ecocode\_groups.abbreviation

(Alternative is to add tbl\_ecocode\_groups.abbreviation)

Single group for BugsEcoCodes needs creation in tbl\_ecocode\_groups.

# TEcoBugs

Mapping

CODE => tbl\_taxonomic\_order.code = > tbl\_ecocodes.taxon\_id

BugsEcoCode => lookup tbl\_ecocode\_definition.abbreviation => tbl\_ecocodes.ecocode\_definition\_id

requires

bugsecodefinitions

# TEcoDefBugs

Mapping

BugsEcoCODE => tbl\_ecocode\_definitions.abbreviation

EcoLabel => tbl\_ecocode\_definitions.label

Definitions => tbl\_ecocode\_definitions.definition ???

Notes => tbl\_ecocode\_definitions.notes ???

SortOrder => tbl\_ecocode\_definitions.sort\_order

lookup tbl\_ecocode\_groups.lable = ‘Bugs group’ => tbl\_ecocode\_definitions.ecocode\_group\_id

# TRDB

Mapping

CODE => lookup tbl\_taxonomic\_orders.code => tbl\_rdb.taxon\_id

CountryCode => lookup tbl\_locations.location\_name type = ‘Country’ => tbl\_rdb.location\_id

RDBCode => lookup tbl\_rdb\_codes.rdb\_code\_id => tbl\_rdb.rdb\_code\_id

convert country code

requires

species + taxonomic order

rdbcodes

countries

# TRDBCodes

Mapping

RDBCode => tbl\_rdb\_codes.rdb\_code\_id ??

Category => tbl\_rdb\_codes.rdb\_category

RDBDefinition => tbl\_rdb\_codes.rdb\_definition

RDBSystemCode => tbl\_rdb\_codes.rdb\_system\_id

requires

rdbsystems

# TRDBSystems

Mapping

RDBSystemCode => tbl\_rdb\_systems.rdb\_system\_id

RDBSystem => tbl\_rdb\_systems.rdb\_system

RDBVersion => tbl\_rdb\_systems.rdb\_version

Publication Year => tbl\_rdb\_systems.rdb\_system\_date

First Published = tbl\_rdb\_systems.rdb\_first\_published

Ref => lookup tbl\_biblio.bugs\_reference => tbl\_rdb\_systems.biblio\_id

Country/International => lookup tbl\_locations.location\_name and type = ‘Country’ => tbl\_rdb\_systems.location\_id

convert country code

requires

countries

# TMCRSummaryData

Mapping

CODE => tbl\_taxonomic\_orders.code => tbl\_mcr\_summary\_data.taxon\_id

TMaxLo => tbl\_mcr\_summary\_data.tmax\_lo

TMaxHi => tbl\_mcr\_summary\_data.tmax\_hi

TMinLo => tbl\_mcr\_summary\_data.tmin\_lo

TMinHi => tbl\_mcr\_summary\_data.tmin\_hi

TRangeHi => tbl\_mcr\_summary\_data.trange\_hi

TRangeLo => tbl\_mcr\_summary\_data.trange\_lo

COGMidTMax => tbl\_mcr\_summary\_data.cog\_mid\_tmax

COGMidTRange => tbl\_mcr\_summary\_data.cod\_mid\_trange

requires

species + taxonomic orders

# TMCRNames

Mapping

CODE => lookup taxonomic\_orders.code => tbl\_mcr\_names.taxon\_id

MCRNumber => tbl\_mcr\_names.mcr\_number

MCRName => tbl\_mcr\_names.mcr\_species\_name

CompareStatus => tbl\_mcr\_names.comparison\_notes

MCRNameTrim => tbl\_mcr\_names.mcr\_name\_trim

requires

species + taxonomic orders

# TSynonym

Mapping

CODE => lookup taxonomic\_orders.code => tbl\_taxa\_synonyms.taxon\_id

SynGenus => lookup tbl\_taxa\_tree\_genera.genus\_name (bugs order) => tbl\_taxa\_synonyms.genus\_id

???

this needs to be checked.

species association (type synonym of) code = target (official sead),   syngenus +  syn authority = source

create necessary taxon (including tree, as needed). and add species association.

Data extraction and insertion needs to handle this too.

Notes:

In data entry needs to find synonyms so that user knows they are entering a synonym, even though it is master species that is stored.

# TTaxoNotes

Mapping

Code => tbl\_taxonomic\_order.code => tbl\_taxonomy\_notes.taxon\_id

Ref => tbl\_biblio.bugs\_reference => tbl\_taxonomy\_notes.biblio\_id

Data => tbl\_taxonomy\_notes.taxonomy\_notes

requires

species + taxonomic order

# TKeys

Mapping

Code -> tbl\_taxonomic\_order.code => tbl\_text\_identification\_keys.taxon\_id

Ref -> tbl\_biblio.bugs\_reference => tbl\_text\_identification\_keys.biblio\_id

Data -> tbl\_text\_identification\_keys.key\_text

requires

species + taxonomic order

# TDistrib

Mapping

Code -> tbl\_taxonomic\_order.code => tbl\_text\_distribution.taxon\_id

Ref -> tbl\_biblio.bugs\_reference => tbl\_text\_distribution.biblio\_id

Data -> tbl\_text\_distribution.distribution\_text

requires

species + taxonomic order

# TBiblio

Use imported version

# TBiology

Mapping

Code -> tbl\_taxonomic\_order.code => tbl\_text\_biology.taxon.id

ref -> tbl\_biblio.bugs\_reference => tbl\_text\_biology.biblio.id

data -> tbl\_text\_biology.biology\_text

required

species + taxonomic order code

# TSpeciesAssociation

Mapping

Code -> tbl\_taxonomic\_order.code => tbl\_species\_associations.taxon\_id

AssociatedSpeciesCODE -> tbl\_taxonomic\_order.code => tbl\_species\_associations.associated\_taxon\_id

AssociationType -> tbl\_tbl\_species\_association\_types.association\_type\_name => tbl\_species\_associations.association\_type\_id (possible conversion needed)

Ref -> tbl\_biblio.bugs\_reference => tbl\_species\_associations.biblio\_id

on association type does not exist raise error

empty type? => is associated with (conversion)

requires

species + taxonomic order

association types (?)

# TAttributes

Mapping

Code => tbl\_taxonomic\_order.code => tbl\_taxa\_measured\_attributes.taxon\_id

AttribType => tbl\_taxa\_measured\_attributes.attribute\_type

AttribMeasure => tbl\_taxa\_measured\_attributes.measure

Value => tbl\_taxa\_measured\_attributes.data

AttribUnits => tbl\_taxa\_measured\_attributes.attribute\_units

requires

species + taxonomic order

# TFossil

Mapping

CODE => tbl\_taxonomic\_order.code => tbl\_abundances.taxon\_id

SampleCODE => bugs\_import.bugs\_traces.bugsdata => tbl\_analysis\_entities.physical\_sample\_id

Abundance => tbl\_abundances.abundance

lookup tbl\_abundance\_elements.element\_name = ‘MNI’ => tbl\_abundance.abundance\_element\_id

group analysis entities based on dataset + physical sample

analysis entity dataset from countsheet (countsheet code = dataset\_name, sheet-type = data\_type, dataset\_master = ‘bugs database’, method = ‘palaeoentomology’)

dataset datatype = countsheet.sheet type  = tbl\_data\_types.data\_type\_name

requires

samples

index

# TSample

Mapping

CountsheeCODE => bugs\_import.bugs\_traces.bugsdata => tbl\_physical\_samples.sample\_group\_id

RefNrContext => tbl\_physical\_samples.sample\_name

lookup tbl\_sample\_types.type\_name = ‘Unspecified’ => tbl\_physical\_samples.sample\_type\_id

lookup tbl\_alt\_ref\_types.alt\_ref\_type = ‘Other alternative sample name’ => tbl\_physical\_samples.alt\_ref\_type\_id

ZorDepthTop => create tbl\_sample\_dimensions

dimension\_value  = ZorDepthTop

method\_id = tbl\_methods.method\_abbrev\_or\_alt\_name = ‘Depth from datum’, ?? this method points to method group 17 => coordinate and altitude systems

dimension\_id = tbl\_dimensions.dimension\_name = ‘Upper boundary depth from unknown reference’ ?? This dimension is pointing to method group 14 => Size measurement

ZorDepthBottom => create tbl\_sample\_dimensions

dimension\_value  = ZorDepthBottom

method\_id = tbl\_methods.method\_abbrev\_or\_alt\_name = ‘Depth from datum’,

dimension\_id = tbl\_dimensions.dimension\_name = ‘Lower boundary depth from unknown reference’

Ignore the following as no data in Bugs:

X => tbl\_sample\_dimensions.dimension\_value [method\_id = ?, dimension\_id = ?]

Y => tbl\_sample\_dimensions.dimension\_value [method\_id = ?, dimension\_id = ?]

Data in X or Y produce warning.

requires

countsheets

# TCountsheet

Mapping

CountsheeName => tbl\_sample\_groups.sample\_group\_name

SiteCODE => bugs\_import.bugs\_traces.bugsData => tbl\_sample\_groups.site\_id

SheetContext => converted lookup tbl\_sample\_group\_sampling\_contexts.sampling\_context => tbl\_sample\_groups.sampling\_context\_id

SheetType =>

Lookup tbl\_method.method\_name = ‘Temporary record’ => tbl\_sample\_groups.method\_id

empty context generate error.

conversions for context

‘Archaeological contexts’ => ‘Archaeological site’

requires

sites

# TSite

Mapping

SiteName => tbl\_sites.site\_name

Region => tbl\_locations.location\_name + type = ‘Unprocessed Bugs Transfer’ => tbl\_site\_locations.location\_id

Country => tbl\_locations.location\_name + type = ‘Country’ => tbl\_site\_locations.location\_id

NGR => tbl\_sites.national\_site\_identifier

LatDD => tbl\_sites.latitude\_dd

LongDD => tbl\_sites.longitude\_dd

Alt => tbl\_sites.altitude

Interp => tbl\_sites.description

SiteCODE => bugs\_import.bugs\_traces.bugsdata

requires

countries

conversions for region

‘Alpes Maritime’ => ‘Alpes-Maritimes’

‘Ameraliksfjord’ => ‘Ameraliksfjordur’

‘Angermannland’ => ‘Angermanland’

‘Co. Down’ => ‘County Dowm’

‘Co. Louth’ => ‘County Louth’

‘Inverness shire’ => ‘Inverness-shire’

‘Leics.’ => ‘Leicestershire’

‘Møen’ => ‘Møn’

‘Noord Brabant’ => ’Noord-Brabant

‘Noord Holland’ => ‘Noord-Holland’

‘North’ => ‘North Holland’

‘Not located’ => NULL

‘Notts.’ => ‘Nottinghamshire’

‘Ostergottland’ => ‘Ostergotland’

‘Ostobottnia media’ => ‘Ostrobothnia media’

‘Ostrobottnia australis’ => ‘Ostrobothnia australis’

‘Ostrobottnia borealis’ => ‘Ostrobothnia borealis’

‘Ostrobottnia media’ => ‘Ostrobothnia media’

‘Reykjavik’ => ‘Reykjavík’

‘S Uist, Outer Hebrides’ => ‘South Uist, Outer Hebrides’

‘Sjaelland’ => ‘Zealand’

‘Vagsøy’ => ‘Vågsøy’

‘Yorks’ => ‘Yorkshire’

‘Ångermanland’ => ‘Angermanland’

–UPDATE tbl\_bugs\_tsite SET “Country” = ‘United Kingdom’ WHERE “SiteCODE” = ‘SITE000006’;

region = Isle of Man, Jersey, Guernsey => location type = Country

# TSIteRef

Mapping

SiteCODE => bugs\_import.bugs\_traces.bugsdata => tbl\_site\_references.site\_id

Ref => tbl\_biblio.bugs\_reference => tbl\_site\_references.biblio\_id

requires

sites

# TSiteOtherProxies

Mapping

SiteCODE => bugs\_import.bugs\_trace.bugsdata => tbl\_site\_other\_records.site\_id

HasPollen => lookup tbl\_record\_types.record\_type\_name(‘External pollen data’) =>  tbl\_site\_other\_records.record\_type\_id

HasPlantMacro => lookup tbl\_record\_types.record\_type\_name(‘External plant macro data’) =>  tbl\_site\_other\_records.record\_type\_id

HasDiatoms => lookup tbl\_record\_types.record\_type\_name(‘Diatoms’) =>  tbl\_site\_other\_records.record\_type\_id

HasChironomids => lookup tbl\_record\_types.record\_type\_name(‘Chironomids’) =>  tbl\_site\_other\_records.record\_type\_id

HasSoilChemistry => lookup tbl\_record\_types.record\_type\_name(‘Soil chemistry/properties’) =>  tbl\_site\_other\_records.record\_type\_id

HasIsotopes => lookup tbl\_record\_types.record\_type\_name(‘Isotopes’) =>  tbl\_site\_other\_records.record\_type\_id

HasAnimalBones => lookup tbl\_record\_types.record\_type\_name(‘Animal bones’) =>  tbl\_site\_other\_records.record\_type\_id

HasArcheology => lookup tbl\_record\_types.record\_type\_name(‘Other archaeology’) =>  tbl\_site\_other\_records.record\_type\_id

HasMolluscs => lookup tbl\_record\_types.record\_type\_name(‘Molluscs’) =>  tbl\_site\_other\_records.record\_type\_id

bugs columns generate new row

requires

sites

# TDatesRadio

Mapping

DateCode => tbl\_Datasets.dataset\_name

SampleCode => bugs\_import.bugs\_traces.bugsdata => tbl\_analysis\_entities.physical\_sample\_id

LabNr => tbl\_geochronology.lab\_number

Uncertainty => lookup tbl\_dating\_uncertainty.uncertainty = Uncertainty => tbl\_geochronology.dating\_uncertainty\_id

Date => tbl\_geochronology.age

AgeErrorOrPlusError => tbl\_geochronology.error\_older

AgeErrorMinus => tbl\_geochronology.error\_younger

DatingMethod => lookup tbl\_methods.method\_abbrev\_or\_alt\_name = DatingMethod => tbl\_datasets.method\_id

LabId => lookup tbl\_dating\_labs.international\_lab\_id = LabId => tbl\_geochronology.dating\_lab\_id

Notes => tbl\_geochronology.notes

1. create tbl\_datasets

master\_set\_id = lookup tbl\_dataset\_masters.master\_name = ’Bugs database"

data\_type\_id = lookup tbl\_data\_types.data\_type\_name = ‘Undefined other’ ? ???? Should this not be the correct data type, or is that not known?

method\_id = lookup tbl\_methods.method\_abbrev\_or\_alt\_name = DatingMethod

dataset\_name = DateCODE

1. create tbl\_analysis\_entities

physical\_sample\_id = Lookup bugs\_import.bugs\_traces.bugsData contain SampleCODE

dataset\_id = created dataset

1. create tbl\_geochronology

analysis\_entity\_id = created analysi entity

dating\_lab\_id = lookup tbl\_dating\_labs.international\_lab\_id = LabId

lab\_number = LabNr

age = Date

error\_older = AgeErrorOrPlusError

error\_younger = AgeErrorMinus || AgeErrorOrPlusError

notes = Notes

Age -> different precision and scale than rest of application?

Dating material??

How to deal with dating specifications with only a note?

-> fix by conversions.

How to deal with dating specifications without data?

-> Skip completely empty items.

-> function for ignoring specific items (error on empty date).

conversions AgeErrorMinus

Null => AgeErrorOrPlusError

dating methods converted to abbreviations in sead: see mapdatingmethod($1, $2)

dating uncertainty updates

c => Ca.

ca => Ca.

from => From

to => To

LabId transformations

Birmingham => Birm

requires

samples

# TLab

Mapping

LabId => tbl\_dating\_labs.international\_lab\_id

Labname => tbl\_dating\_labs.lab\_name

Country => lookup tbl\_locations.location\_name = Country and type = ‘Country’ => tbl\_dating\_labs.country\_id

# TDatesCalendar

Mapping

SampleCODE => lookup bugs\_import.bugs\_trace.bugsdata => tbl\_relative\_dates.physical\_sample\_id

Uncertainty => lookup tbl\_dating\_uncertainty.uncertainty => tbl\_relative\_dates.dating\_uncertainty\_id

Date + BCADBP => concatenate ‘CAL\_’ + Date + BCADBP => lookup tbl\_relative\_ages.“Abbreviation” => tbl\_relative\_dates.relative\_age\_id

DatingMethod => lookup tbl\_methods.method\_abbrev\_or\_alt\_name => tbl\_relative\_dates.method\_id

Notes => tbl\_relative\_dates.notes

date + bcadbp => new relative age, set age data from values;

   -> relative age type from dating method (?)

How to map dating method to relative age type?

Fix naming before or string lookup?

set method by datingMethod ?

 -> not the same thing

From + to from same sample generates one relative age (start and stop dates). type = calendar date range.

 -> a range is one relative dates also.

Single for a sample => relative age type = calendar date

Normalize date data

convertsions

uncertainty

To > => >

from ca. => From ca.

Tc ca. => To ca.

Ca, => Ca.

requires

samples

# TDatesPeriod

Mapping

PeriodCode => bugs\_import.bugs\_trace.bugsdata => tbl\_relative\_dates.relative\_age\_id

SampleCODE => bugs\_import.bugs\_trace.bugsdata => tbl\_relative\_dates.physical\_sample\_id

Uncertainty => lookup tbl\_dating\_uncertainty.uncertainty => tbl\_relative\_dates.dating\_uncertainty\_id

DatingMethod => lookup tbl\_methods.metod\_abbrev\_or\_alt\_name => tbl\_relative\_dates.method\_id

Notes => tbl\_relative\_dates.notes

requires

periods

samples

transformations

dating method = ‘ArchPer’ and period.yeartype = ‘C14’ => method name = ArchPerC14

dating method = ‘ArchPer’ and period.yeartype = ‘Calendar’ => method name = ArchPerCal

dating method = ‘GeolPer’ and period.yeartype = ‘C14’ => method name = GeolPerC14

dating method = ‘GeolPer’ and period.yeartype = ‘Calendar’ => method name = GeolPerCal

in script ’ ’ (single space) uncertianty is ok, but cannot be read using external library, or not be stored via newer Access. Cannot test this but will build into functionality.

# TPeriods

Mapping

Period Name => tbl\_relative\_ages.relative\_age\_name

Type => lookup tbl\_relative\_age\_types.age\_type = Type => tbl\_relative\_ages.relative\_age\_type\_id

Description => tbl\_relative\_ages.description

Reference => lookup tbl\_biblio.bugs\_reference => tbl\_relative\_age\_refs.biblio\_id

Geography => lookup tbl\_locations.location\_name => tbl\_relative\_ages.location\_id

Begin + BP etc => convert using type = ‘begin’ and method = ‘C14’ => tbl\_relative\_ages.c14\_age\_older

End + BP etc => convert using type = ‘end’ and method = ‘C14’ => tbl\_relative\_ages.c14\_age\_younger

Begin + BP etc => convert using type = ‘begin’ and method = ‘cal’ => tbl\_relative\_ages.cal\_age\_older

End + BP etc => convert using type = ‘end’ and method = ‘cal’ => tbl\_relative\_ages.cal\_age\_younger

PeriodCODE => tbl\_relative\_ages.“Abbreviation”

exclude for periodcode = ‘?’

conversion for age values

(see ImportFunctions.GetPeriodAge(date\_value, bcad\_value, age\_direction, age\_type))

-> if type = c14 convert iff bcad\_value = ‘BP’

-> if type = ‘cal’ convert iff bcad\_value = ‘BP’

usage:

GetPeriodAge(prow.periodbegin, prow.beginbcad, ‘beginbcad’, ‘c14’) => c14\_age\_older

GetPeriodAge(prow.periodend, prow.endbcad, ‘endbcad’, ‘c14’) => c14\_age\_younger

GetPeriodAge(prow.periodbegin, prow.beginbcad, ‘beginbcad’, ‘cal’) => cal\_age\_older

GetPeriodAge(prow.periodend, prow.endbcad, ‘endbcad’, ‘cal’) => cal\_age\_younger

reduce this…

date\_value = null => null

bcad\_value = null & date\_value = 0 & age\_type = ‘c14’ => 0

bcad\_value = null & date\_value = 0 & age\_type != ‘c14’ => null

bcad\_value = ‘BP’ & age\_type = ‘cal’ => return null;

bcad\_value = ‘BP’ & age\_type != ‘cal’ & age\_direction = ‘beginbcad’ => date\_value

bcad\_value = ‘BP’ & age\_type != ‘cal’ & age\_direction = ‘endbcad’ => date\_value

(bcad\_value = ‘AD’ or bcad\_value = ‘BC’) & age\_type = ‘c14’ => null

bcad\_value = ‘AD’ & age\_direction = ‘beginbcad’ => 1950 - date\_value

bcad\_value = ‘AD’ & age\_direction = ‘endbcad’ => 1950 - date\_value

bcad\_value = ‘BC’ & age\_direction = ‘beginbcad’ => date\_value + 1950

bcad\_value = ‘BC’ & age\_direction = ‘endbcad’ => date\_value + 1950

Abbreviation column name should be changed!!!!

Note:

Data cleaning in Bugs needed… still