Admin Dokumentation: PostgreSQL Snippets

Recherche

Nachfolger in aktueller Kartierung finden

Suche nach Giscode eines alten Biotops in Kartierobjekte (Vorgänger oder Zusammengefasste Bögen) bzw. Verlustbogen.

```
SELECT * FROM mvbio.kartierobjekte
WHERE 119474 IN (SELECT UNNEST(zusammengefasste_ids) UNION SELECT
vorgaenger_id)
```

bzw. in Verlustbögen:

```
SELECT * FROM mvbio.verlustobjekte
WHERE bogen_id IN (SELECT id FROM archiv.erfassungsboegen WHERE id=119474)
```

Artenvorkommen

Suche nach Biotopen/Bögen, die eine bestimmte Art (Beispiel hier: Krebsschere *Stratiotes aloides*) enthalten. Mit Angabe der Häufigkeit (D/Z/V) und Geometriespalte (Well-Known-Text WKT, einfacher Import in QGIS möglich). Anpasssung der Art in WHERE-Klausel, auch nur Gattung möglich (Artname wird mit ausgegeben).

Aktuelle Kartierung

Ohne Einschränkung der Bearbeitungsstufe!

```
SELECT
  gsl.valid_name AS art_gsl, pv.dzv AS dzv, ko.id AS id_mvbio, ko.label,
ko.biotopname, ko.vegeinheit,
  ko.hc, COALESCE(ko.uc1, '-') || ' ' || COALESCE(ko.uc2, '-') AS ucs,
  ko.e_datum AS datum, ko.kartierer, kk.bezeichnung AS quelle,
  ST_AsEWKT(ko.geom) AS geom_wkt
FROM mvbio.kartierobjekte ko
JOIN mvbio.pflanzenvorkommen pv ON pv.kartierung_id = ko.id
LEFT JOIN mvbio.kampagnen kk ON kk.id = ko.kampagne_id
LEFT JOIN mvbio.pflanzenarten_gsl gsl ON gsl.valid_nr = pv.valid_nr
WHERE
  gsl.valid_name LIKE 'Stratiotes%' AND gsl.synonym='0' AND kk.art IN
('BK','LRT','BLRTK')
ORDER BY
  art_gsl, CASE dzv WHEN 'D' THEN 1 WHEN 'Z' THEN 2 WHEN 'V' THEN 3 END
```

Archivbestände

nur aktuelle Bögen

```
SELECT
  gsl.valid_name AS art_gsl, pv.dzv AS dzv, eb.id AS id_archiv, eb.label,
eb.biotopname, eb.vegeinheit AS vegeinheit, eb.hc,
  COALESCE(eb.uc1, '-') || ' ' || COALESCE(eb.uc2, '-') AS ucs,
  eb.e_datum AS datum, eb.kartierer, kk.bezeichnung AS Quelle,
  ST_AsEWKT(eb.geom) geom_WKT
FROM archiv.erfassungsboegen eb
JOIN archiv.pflanzenvorkommen pv ON pv.kartierung_id = eb.id
LEFT JOIN archiv.kampagnen kk ON kk.id = eb.kampagne_id
LEFT JOIN mvbio.pflanzenarten_gsl gsl ON gsl.valid_nr = pv.valid_nr
WHERE
  gsl.valid_name LIKE 'Stratiotes%' AND gsl.synonym='0' AND eb.aktuell
ORDER BY
  art_gsl, CASE dzv WHEN 'D' THEN 1 WHEN 'Z' THEN 2 WHEN 'V' THEN 3 END
```

Biotopcodes

Suche aller Biotopbögen, die in Haupt-, Neben- oder Überlagerungscode einen bestimmten Biotopcode enthalten. Abfrage mehrere Biotopcodes möglich. Nutzt die LINFOS-Export Views, ersetzt aber die Spalte geom durch WKT.

im Archiv

```
SELECT id_mvbio, giscode, obj_code, kampagne, kartgeb, kartebene, bogenart,
kart_jahr, aktuell, biotopname, area_ha, bntk_code, hc, hcnum, hcarea,
hcname, nc1, nc1num, nc1area, nc1name, nc2, nc2num, nc2area, nc2name, nc3,
nc3num, nc3area, nc3name, nc4, nc4num, nc4area, nc4name, nc5, nc5num,
nc5area, nc5name, nc6, nc6num, nc6area, nc6name, nc7, nc7num, nc7area,
nc7name, nc8, nc8num, nc8area, nc8name, uc1, uc1name, uc2, uc2name, n_foto,
erfasser, abfrage_datum, ST_AsEWKT(geom) AS geom_wkt
FROM archiv.v_linfos_aktuelle_biotope
WHERE ARRAY[hc,nc1,nc2,nc3,nc4,nc5,nc6,nc7,nc8,uc1,uc2] &&
ARRAY['XGM','XGW','XGL','XGT','UHL']::VARCHAR[]
```

in aktueller Kartierung

```
SELECT obj_code, id_mvbio, kampagne, kartgeb, kartebene, bogenart, ffh,
kart_jahr, biotopname, lrt_code, lrt_nummer, lrt_bez, e_zustand, e_gutacht,
area_ha, hc, hcnum, hcarea, hcname, nc1, nc1num, nc1area, nc1name, nc2,
nc2num, nc2area, nc2name, nc3, nc3num, nc3area, nc3name, nc4, nc4num,
nc4area, nc4name, nc5, nc5num, nc5area, nc5name, nc6, nc6num, nc6area,
nc6name, nc7, nc7num, nc7area, nc7name, nc8, nc8num, nc8area, nc8name, uc1,
```

```
uc1name, uc2, uc2name, n_foto, erfasser, lrt_kat, lrt_klas, legende, stand,
abfrage_datum, ST_AsEWKT(geom) AS geom_wkt
FROM mvbio.linfos_export
WHERE ARRAY[hc,nc1,nc2,nc3,nc4,nc5,nc6,nc7,nc8,uc1,uc2] &&
ARRAY['XGM','XGW','XGL','XGT','UHL']::VARCHAR[]
```

Qualitätskontrolle

Verlustbögen: Herkunft des untergegangenen Bogens

Verlustmeldungen dürfen sich nur auf aktuelle Biotope (BK1/BK1315, ...) beziehen.

Abfrage, welche Verlustbögen sich nicht auf einen aktuelles Biotop beziehen:

```
SELECT
 v.id, a.giscode, v.kartierer, v.bearbeitungsstufe,
 kk.abk, a.aktuell
FROM mvbio.verlustobjekte v, archiv.erfassungsboegen a
JOIN archiv.kampagnen kk ON kk.id = a.kampagne_id
WHERE v.bogen_id = a.id  AND v.kartiergebiet_id != 3459 AND (NOT a.aktuell
OR a.kampagne_id NOT IN (1,3,12,15))
ORDER BY a.kampagne_id
```

Vollständigkeit Differenzflächen (BK2021)

Abfrage listet alle Differenzflächen auf, die nicht in einem Kartier- oder Verlustobjekt geanannt werden. + Link zum BK1-Bogen in MVBIO-Pro.

Die Spalten biotop_inters bzw. verlust_inters zeigen an, ob eine räumliche Überlagerung mit Kartierobjekten vorliegt.

```
WITH diff AS
  (SELECT df.giscode giscode_diff, df.los_nr,
    EXISTS (SELECT * FROM mvbio.kartierobjekte ko WHERE
st_intersects(ko.geom,df.the_geom)) AS biotop_inters,
    EXISTS (SELECT * FROM mvbio.kartierobjekte ko WHERE
ko.giscode=df.giscode OR df.giscode IN (SELECT UNNEST
(ko.zusammengefasste_boegen))) AS biotop_giscode,
    EXISTS (SELECT * FROM mvbio.verlustobjekte vo WHERE
st_intersects(vo.geom,df.the_geom)) AS verlust_inters,
    EXISTS (SELECT * FROM mvbio.verlustobjekte vo WHERE
vo.giscode=df.giscode) AS verlust_giscode
   FROM mvbio.differenzflaechen df)
SELECT * ,
 CASE
  WHEN SUBSTRING(giscode_diff FROM 10 FOR 1)='4' THEN
((('=HYPERLINK("https://mvbio.de/kvwmap/index.php?go=Layer-Suche_Suchen&sele
```

```
cted_layer_id=140&value_label='::text || giscode_diff) ||
'&operator_label=%20=";"'::text) || giscode_diff::text) || '")'::text
  ELSE
((('=HYPERLINK("https://mvbio.de/kvwmap/index.php?go=Layer-Suche_Suchen&sele
cted_layer_id=196&value_label='::text || giscode_diff) ||
'&operator_label=%20=";"'::text) || giscode_diff::text) || '")'::text
 END AS giscode_link
FROM diff
WHERE NOT biotop_giscode AND NOT verlust_giscode AND los_nr=2 ORDER BY
biotop_inters, giscode_diff
```

Rückgabewünsche zurücksetzen

Setzt alle Bögen mit Rückgabewunsch in allen Kampagnen auf Stufe 2 zurück und ergänzt eine Info in den Prüfhinweisen.

```
UPDATE mvbio.kartierobjekte
SET
pruefer_rueckweisung=TRUE, --deaktiviert die Prüfung
"validate_kartierobjekte"
pruefer_pruefhinweis=COALESCE(pruefer_pruefhinweis||',
','')||'['||CURRENT_DATE::text||']: Rückgabewunsch',
bearbeitungsstufe=2
WHERE rueckgabewunsch

--analog für Verlustbögen:
UPDATE mvbio.verlustobjekte
SET
pruefer_rueckweisung=TRUE, --deaktiviert die Prüfung
"validate_kartierobjekte"
pruefer_pruefhinweis=COALESCE(pruefer_pruefhinweis||',
','')||'['||CURRENT_DATE::text||']: Rückgabewunsch',
bearbeitungsstufe=2
WHERE rueckgabewunsch
```

Rückgabe von Bögen mit QS-Fehlern

Genutzt wird der View mvbio.qs_00_alle_boegen, der die IDs aller Bögen aus den QS-Views sammelt. Im Prüfkommentar werden die QS-Codes mit Datum angehängt. Nach Rücksetzung sind die QS-Views und mvbio.qs_00_alle_boegen wieder leer. Analog für Verlustbögen

```
UPDATE mvbio.kartierobjekte k
SET
LOCK = TRUE, --deaktiviert die Prüfung "validate_kartierobjekte"
pruefer_rueckweisung=TRUE,
pruefer_pruefhinweis=COALESCE(pruefer_pruefhinweis||',
','')||'['||CURRENT_DATE::text||']: QS '||COALESCE((SELECT fehlercodes FROM
mvbio.qs_00_alle_boegen qs WHERE k.id=qs.bogen_id),'alles ok'),
```

```
bearbeitungsstufe=2
WHERE id IN (SELECT bogen_id FROM mvbio.qs_00_alle_boegen);

--analog für Verlustbögen:
UPDATE mvbio.verlustobjekte vo
SET
LOCK = TRUE, --deaktiviert die Prüfung "validate_kartierobjekte"
pruefer_rueckweisung=TRUE,
pruefer_pruefhinweis=COALESCE(pruefer_pruefhinweis||',
','')||'['||CURRENT_DATE::text||']: QS '||COALESCE((SELECT fehlercodes FROM
mvbio.qs_00_alle_verluste vq WHERE vo.id=vq.verlust_id),'alles ok'),
bearbeitungsstufe=2
WHERE id IN (SELECT verlust_id FROM mvbio.qs_00_alle_verluste);
```

Datenbankstruktur

Tabellen-Information

Liste aller Spalten einer Datenbank-Tabelle mit Datentyp, Länge und Beschreibung (Comment). Funktioniert auch für mehrere Tabellen (z.B. alle eines Schemas), dafür die WHERE Klausel anpassen.

```
SELECT
   c.table_schema, c.table_name, c.column_name, c.data_type,
   COL_DESCRIPTION(CONCAT(c.table_schema, '.', c.table_name)::regclass,
c.ordinal_position) AS description,
COALESCE(c.character_maximum_length::text,c.numeric_precision::text||'('||c.
numeric_scale::text||')') AS column_length
FROM information_schema.columns AS c
JOIN information_schema.tables AS t
   ON t.table_catalog = c.table_catalog
   AND t.table_schema = c.table_schema
   AND t.table_name = c.table_name
WHERE  t.table_type = 'BASE TABLE'
   AND c.table_schema = 'mvbio'
   AND c.table_name ='kartiergebiete'
ORDER BY c.table_schema, c.table_name, c.ordinal_position;
```

Tabellenvergleich

Vergleicht zwei Tabellen, für Gegenüberstellung archiv ↔ mvbio (Spaltenname, Typ, Comment). Im Feld `data_typ` werden unterschiedliche Datenypen mit *!!!* hervorgehoben. Zieltabellen in der `WITH` Definition anpassen.

```
WITH
  kga AS (SELECT * FROM information_schema.columns WHERE table_schema =
'archiv' AND TABLE_NAME = 'kartiergebiete' ),
  kg AS (SELECT * FROM information_schema.columns WHERE table_schema =
```

```
'mvbio' AND TABLE_NAME = 'kartiergebiete')
SELECT
  kg.column_name AS mvbio_name,
  kga.column_name AS archiv_name,
  CASE
    WHEN kg.data_type=kga.data_type THEN kga.data_type
    WHEN kg.data_type IS NULL OR kga.data_type IS NULL THEN
COALESCE(kg.data_type,kga.data_type)
    ELSE '!!! '||kg.data_type||' vs. '||kga.data_type
  END AS data_typ,
  CASE
    WHEN kg.table_schema IS NOT NULL THEN
COL_DESCRIPTION(CONCAT(kg.table_schema, '.',
kg.table_name)::regclass,kg.ordinal_position)
    ELSE NULL
  END AS mvbio_desc,
  CASE
    WHEN kga.table_schema IS NOT NULL THEN
COL_DESCRIPTION(CONCAT(kga.table_schema, '.',
kga.table_name)::regclass,kga.ordinal_position)
    ELSE NULL
  END AS archiv_desc
FROM
 kg
FULL JOIN kga ON kg.column_name=kga.column_name
ORDER BY kg.ordinal_position
```