CAD-Dokumentation zu GIS mit SpatiaLite migrieren

Christoph Rinne

09. Januar 2023

Inhaltsverzeichnis

Vorwort

1 Einführung

Ziel ist die Überführung von Ausgrabungsplänen aus CAD-Dateien in ein GIS. Ausgang ist der CAD-Plan einer Ausgrabung für ein Landesdenkmalamt über insgesamt drei Ausgrabungsphasen mit einem Personalwechsel bei der Ausgrabungsleitung und der Grabungstechnik [1]. Der dargestellte Weg ist einer von vielen möglichen Wegen.

Anmerkungen

- Menüpfade oder Abfolgen von Fenstern werden mit schlichten Pfeilen dargestellt: "Datei > Speichern".
- Tastaturkürzel, die ich gerne Nutze, stehen in Spitzklammern je Taste: <strg> + <c>.
- Schalter auf Formularen werden in [] gesetzt: [OK]
- Zur Darstellung von Befehlen im Text nutze ich die in Markdown übliche Darstellung von Code oder eben Anweisungen an den Computer: `anweisung`.
- SQL-Anweisungen sind nicht in Großbuchstaben gesetzt, die farbliche Gestaltung macht dies überflüssig. Eine Ausnahme bilden die verwendeten Funktion bei denen der *CamelCase* für die bessere Lesbarkeit beibehalten wird. Auch wird der Anfang einer Anweisungen jeweils großgeschrieben, um aufeinander folgende Anweisungen etwas besser zu trennen. Im vorliegenden Fall wurden Leerzeichen in Objektnamen vermieden, dadurch müssen Tabellen und Feldnamen nicht in " " stehen.
- Der Text enthält viele Links die auf Papier nicht funktionieren. Sparen Sie bitte Papier und verzichten Sie auf den Ausdruck.

1.1 Verwendete Software & Informationen

- OS Windows 10
- QGIS 3.22.4-Białowieża Quelle: [https://qgis.org]
- SpatiaLite SpatiaLite GUI 2.1.0 beta1, SpatiaLite 5.0.0, SQLite 3.33.0, Quelle [http://www.gaia-gis.it]
- AutoCAD 2010, Quelle für aktuelle *kostenlose* Schulversionen: [https://www.autodesk.de/education/edu-software/overview]
- SpatiaLite Cookbook html [http://www.gaia-gis.it/gaia-sins/spatialite-cookbook/index.html]
- SpatiaLite Funktionen [http://www.gaia-gis.it/gaia-sins/spatialite-sql-5.0.0.html]

AutoCAD ist eine sehr komplexe Software und Ausgrabungen können eine komplexe Struktur annehmen, die es zu dokumentieren gilt. Erwarten Sie nicht, dass die notwendige Kompetenz beim Erstellen der digitalen Daten stets vorhanden war, auch der Autor (Chr. Rinne) ist hier nur Autodidakt.

Rechnen Sie mit Fehlern im originalen Datenbestand und einer ggf. nicht optimalen Struktur oder erwarten Sie auch nicht die von Ihnen bevorzugte Struktur. Korrektur von Fehler und Anpassungen der Struktur erfolgen sicher am besten im originalen Arbeitsumfeld, also CAD.

Neben AutoCAD gibt es teils kostengünstigere Alternativen, u.a.:

- BricsCAD [https://www.bricsys.com]
- MegaCAD [https://www.megacad.de/]

2 Jakob-Uffrecht-Straße

2.1 AutoCAD Quelldatei (dxf)

Es handelt sich um einen mehrperiodigen Siedlungs- und Bestattungsplatz. Untersucht wurden gut 8.300 m² mit 589 Befunden, überwiegend der Bronzezeit (316), 11 eindeutig neolithischen Befunden, darunter ein doppeltes Grabensystem, 260 nicht weiter datierten oder allgemein urgeschichtlichen und zwei neuzeitlichen Befunden. Zu dem CAD Plan liegen für jede Ausgrabung eine Datenbank (MS Access)

[1] Genehmigung des Amtes und der Ausgräberin noch erbitten.

mit weiteren Informationen vor. Diese Daten können nach der Aufarbeitung des CAD Planes mit den dann eindeutig benannten Befunden verbunden werden. Dies ist aber nicht Teil dieses Skriptes.

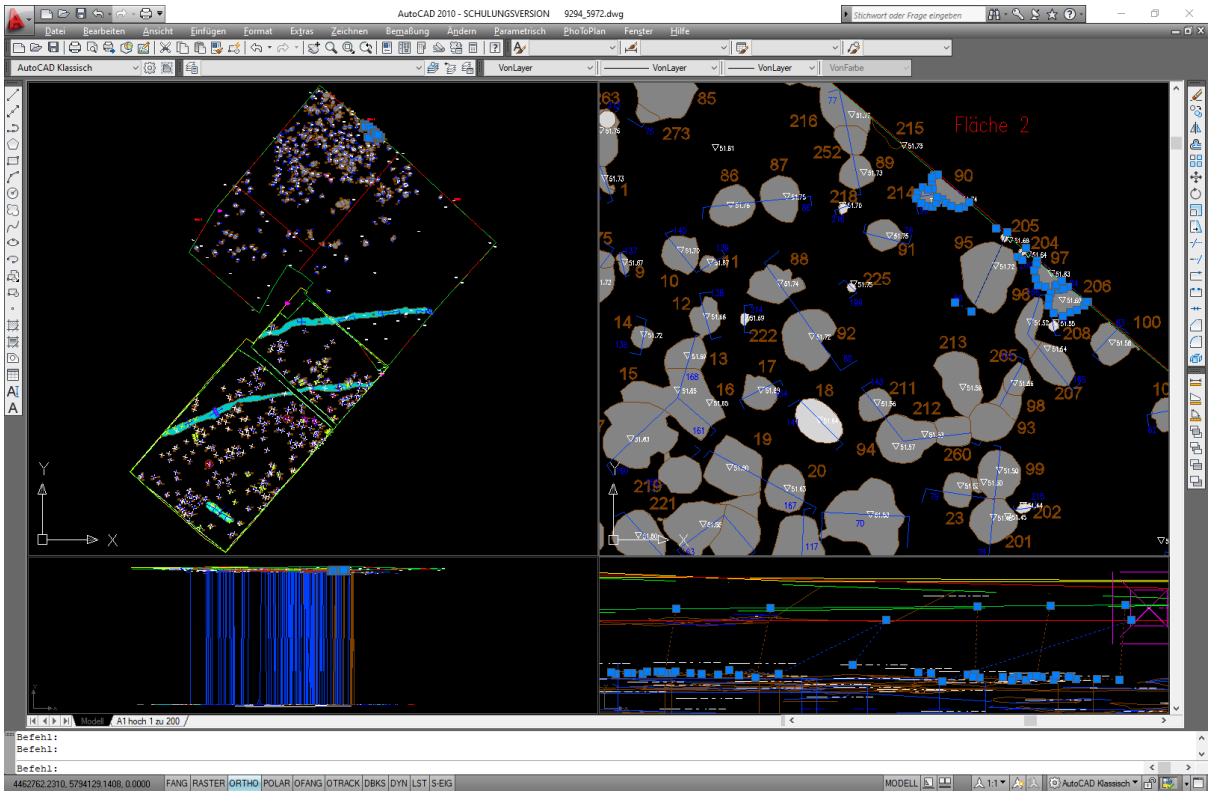

Abbildung 1: Abb. 1 CAD-Datei Jacob-Uffrecgt-Straße in AutoCAD

Die Zeichnung enthält 171 Polylinien (2D), 33 Kreise (2D), 263 Absatztexte (MText), 361 Linien (3D), 239 Schraffuren, 427 Blockreferenzen, 810 Polylinien (3D), 497 einfache Texte und 369 Punkte (3D). In dieser Liste fallen vor allem zwei Objektypen auf, die Kreise und der Absatztext. Die Kreise wurden für einige Befunde verwendet und liegen als flache Geometrie auf einer sinnvollen Höhe. Der Absatztext wurde bei einer Maßnahme für die Befundnummern verwendet. Da es sich um keine "normale" Geometrie handelt und nur die Textbox aber nicht der Text einen Lagebezug zu den Koordinaten der Grabung hat muss dies noch in AutoCAD aufgelöst werden. Ebenfalls interessant, auch mit Blick auf die Umsetzung in SpatiaLite, sind als 2D-Polylinien mit Erhebung vorliegende Befunde. Nutzen Sie `_qselect` mit den Optionen Objekttyp: Polylinie und der Eigenschaft Layer=Befund, um diese pauschal auszuwählen und mit einer Farbvorgabe, z.B. Magenta, deutlich hervorzuheben.

Name	Farbe	Linientyp	Plotstil
0	weiß	Continuous	Color_7
00Befund	36	Continuous	Color_36
00BefundNr	36	Continuous	Color_36
00Profil	blau	Continuous	Color_5
00ProfilNr	blau	Continuous	Color_5
Befunde	36	Continuous	Color_36
Befunde Planum 2	33	Continuous	Color_33
Befundnummern	36	Continuous	Color_36
Bronzenadel	124	Continuous	Color_124
CONTROL	magenta	Continuous	Color_6
Defpoints	weiß	Continuous	Color_7
Flächengrenzen	rot	Continuous	Color_1
GGok	50	Continuous	Color_50
GGuk	weiß	Continuous	Color_7
Grabungsgrenze	102	STRICHPUNKTX2	Color_102

Name	Farbe	Linientyp	Plotstil
Gruben	252	Continuous	Color_252
GUK	90	ACAD_ISO10W100	Color_90
HÖHEN	weiß	Continuous	Color_7
Höhenpunkte Befunde	weiß	Continuous	Color_7
Keramik Pl.1	60	Continuous	Color_60
Knochenreste	40	Continuous	Color_40
Neolithikum	120	Continuous	Color_120
Papierbereich	weiß	Continuous	Color_7
Papierbreich Legende	weiß	Continuous	Color_7
Passpunkte	magenta	Continuous	Color_6
Pfosten	254	Continuous	Color_254
Photogrammetriepunkte	160	Continuous	Color_160
Pl1-Keramik	rot	Continuous	Color_1
Pl2-Befunde	weiß	Continuous	Color_7
Pl2-Keramik	rot	Continuous	Color_1
Pl2-Knochen	blau	Continuous	Color_5
Profile	160	Continuous	Color_160
Profilzeichennägel	200	Continuous	Color_200
Salzmünder Kultur	132	Continuous	Color_132
SBZ EZ	16	Continuous	Color_16
Silex-Messer	220	Continuous	Color_220
Störung	250	Continuous	Color_250
Umrandung Hausgrundriss	blau	Continuous	Color_5

Eine weitere Kontrolle ergibt:

- Die Anzahl der Layer ist überschaubar, eine Trennung nach den Grabungsflächen ist nicht erfolgt, die Namen sind leider nicht ganz stringent, vor allem die Befundnummern sind mehrfach vertreten.
- Einige Namen verweisen auf konkrete Objekte und entsprechen damit nicht der scheinbar allgemein angewendeten Nomenklatur.
- Es sind keine weiteren BKS (Benutzerkoordinatensysteme) definiert, die Koordinaten lassen ein GKB Zone 4 vermuten (das Elipsoidmodell ergibt sich daraus aber nicht).
- Die Einheit ist erwartbar in Millimeter statt den verwendeten Meter.
- Zahlreiche Befundlinien sind nicht geschlossen, auf dem Befundlayer liegen auch Kreise.
- Die Befundnummern sind über die einzelnen Ausgrabungen nicht fortlaufend vergeben, es gibt somit doppelte Befundnummern in der nördlichen und südlichen Fläche.
- Die Ansicht von der Seite offenbart das größte Problem. Vielfach laufen Polygone von der realen Höhe auf 0 runter. Daneben sind aber auch in der Grabung mehrere Höhenbereiche der Objekte mit einzelnen Verbindungslinien zu erkennen. Falsche Prismenhöhe beim Messen?

Es ist vor allem dieses letzte Problem, das bei der nachfolgenden Aufarbeitung besonders betrachtet werden soll.

Wichtig Der MText wird weder von SpatiaLite noch von QGIS beim Import der DXF-Datei erkannt. Der Export dieser Objekte in AutoCAD über "Extras > Datenextraktion" ist möglich, der Inhalt wird aber auf den Ursprung der Textbox bezogen und die ggf. mehrzeiligen Texte werden als ein Textfeld hieran angehängt wodurch sich jede Zeile in Abhängigkeit der Texthöhe zunehmend von der Koordinate entfernt. Es empfiehlt sich, MText mit dem Befehl `ursprung (_explode)` in einfachen Text aufzulösen. Damit wird er auch beim Import der DXF-Datei je Zeile als Text erkannt und dem Einfügepunkt, meist der linke Ursprung der Basislinie, zugewiesen.

Um MText pauschal in Text umzuwandeln selektieren sie diesen pauschal mit `qselect`, Anwenden auf: Ganze Zeichnung, Objekttyp: MText, Eigenschaft: Farbe = VonLayer, "In neuen Auswahlsatz einfügen" und [OK]. Danach den Befehl `ursprung` für die ausgewählten Objekte eingeben und ausführen. Da unter Umständen MText nicht die Farbe des Layers gehabt haben kann sollten Sie erneut alles Markieren und im Fenster der Eigenschaften ("eigenschaften"') im oberen drop-down die Anzahl der vorhandenen Objekttypen auf MText kontrollieren.

2.2 SpatiaLite GUI

Nach dem Start der GUI bitte "Menu > Create a new (empty) SQLite DB" ausführen. In diese wird die dxf-Datei mit der gesamten Ausgrabung importiert: "Menu > Advanced > Import DXF drawings". Im Fenster zum Import bitte folgende Angaben: (x) Import selected DXF drawing file only, SRID: 31467, [v] Append to already existing tables, Dimensions: (x) authomatic 2D/3D, (x) mixed layers, Special Rings handling (x) none. Diese Angaben beziehen sich auf den aktuellen Import und müssen ggf. angepasst werden. Die Option 'mixed layers' trennt nur die Typen (Punkt, Linie Polygon, Text), die Layer werden als Attribut (Spalte) angelegt. Die Alternative trennt erst die Layer und dann nach Typen, erstellt also das x-Fache der vorhandenen CAD-Layer als Tabellen.

Als Ergebnis sind folgende Tabellen mit den Spalten *feature_id*, *filename*, *layer* und *geometry* vorhanden:

- hatch_layer_2d
- hatch_layer_2d_pattern
- line_layer_3d
- point_layer_3d
- polyg_layer_3d
- text_layer_3d

2.2.1 Datenkontrolle

Visualisieren Sie den Inhalt der Geometrietabellen in QGIS. Die Option "Map preview" im Kontextmenü zu jeder Geometriespalte der Tabellen in SpatiaLite ist oft hilfreich, bietet aber weniger Möglichkeiten. Ergänzend wird in SpatiaLite für jede Tabelle eine zusammenfassende Übersicht der Daten generiert.

Die Tabelle **hatch_layer_2d** scheint ein vollständiges(?) Abbild der dokumentierten Befunde. Sie resultiert aus den Schraffuren zur Datierung oder Befundansprache, die auf jeweils eigenen Layern in AutoCAD abgelegt wurden. Diese Informationen sind nett, sollten aber als Sachinformationen später aus den vorhandenen Datenbanken an die Befundnummern angehängt werden (join). Die Geometrie ist ein Multipolygon, einige sind fehlerhaft (NULL).

```
Select layer, GeometryType(geometry) as geomtype, Count(*) as n
from hatch_layer_2d
group by 1, 2;
```

Die Tabelle **hatch_layer_2d_pattern** sieht ähnlich aus, liefert aber nur zwei Geometrien, die überwiegenden Fälle sind NULL.

In AutoCAD bestehen **Füllungen** (Pattern) wie bei vielen Grafikprogrammen aus zwei Elemente: der Kontur (Umgrenzung) und der Füllung. AutoCAD verwendet überwiegend Schraffuren, teils komplexe Kombinationen aus Linien, Polygonen und Punkten. Daneben gibt es die Füllung "solid", die eher eine Ausnahme darstellt. Erstere finden sich in der Tabelle mit mit der Ergänzung "_pattern"- letztere in der Tabelle ohne die Extension. Schraffuren sind in AutoCAD Flächen und damit sind diese Geometrien auch hier 2D.

- Die vorgenannten Tabellen sind damit weitgehend wertlos, denn die 2D Geometrie ist ohne z-Wert unzureichend und die Sachinformationen stehen in der zugehörigen Datenbank des Projektes.

Die Tabelle **line_layer_3d** ist von zentraler Bedeutung und zeigt außer der nicht stringenten Benennung der Layer keine Auffälligkeiten. Allerdings irritiert der mangelhafte Bezug der Profile zu den Layern (00Profil, Profile). Zudem sind offenbar zahlreiche Befunde nur als Linien erkannt worden. Die Profile weisen mit "Ansichtshaken" auf die jeweils dokumentierten Seite hin.

```
select layer, GeometryType(geometry) as geomtype, Count(*) as n
from line_layer_3d
group by 1, 2;
```

Die Tabelle **point_layer_3d** enthält 3D-Punkte, die durch die Layerbezeichnungen teilweise von Bedeutung sind. Überwiegend sind es Höhenwerte der einzelnen Befunde die aber keinen Planumsbezug oder benannten Befundbezuge haben und mit den vorliegenden z-Polygonen für die Befunde überflüssig sind. Von interesse erscheinen die Verweise auf Funde (Keramik, Knochen) und Messbildnägel für die Photgrammetrie.

```
select layer, GeometryType(geometry) as geomtype, Count(*) as n
from point_layer_3d
group by 1, 2;
```

Die Tabelle **polyg_layer_3d** ist von zentraler Bedeutung. Es handelt sich überwiegend um Befunde und einige Flächengrenzen auf unterschiedlichen Layern. Irritierend ist ein Profilpolygon und ein als Polygon dokumentiertes Keramikfargment. Bei ersterem handelt es sich um zwei, gemeinsam als Kasten ausgehobene Profile zu zwei dicht beieinander liegenden Befunden.

```
select layer, GeometryType(geometry) as geomtype, Count(*) as n
from polyg_layer_3d
group by 1, 2;
```

Die Tabelle **text_layer_3d** hat zu den oben genannten Spalten noch *label* und *rotation*. Es sind überwiegend die Befund- und Profilnummern, dazu einige weitere Beschriftungen. Die im CAD-Plan als Block mit Attribute gesetzten Niv-Werte sind nicht dabei und müssen über die Datenextraktion und die resultierende CSV-Liste wieder importiert werden.

```
select layer, GeometryType(geometry) as geomtype, Count(*) as n
from text_layer_3d
group by 1, 2;
```

2.2.2 Topologie prüfen und reparieren

Zur Datenbereinigung werden in allen Geometriespalten pauschal ungültige Geometrien repariert.

```
Begin transaction;
Update line_layer_3d
 set geometry = SanitizeGeometry(geometry);
Update point_layer_3d
 set geometry = SanitizeGeometry(geometry);
Update polyg_layer_3d
 set geometry = SanitizeGeometry(geometry);
Update text_layer_3d
 set geometry = SanitizeGeometry(geometry);
Commit;
```

In der Tabelle **line_layer_3d** bleiben vier Fälle, die nur aus zwei identischen Punkten bestehen. Diese sind schlicht wertlos und werden gelöscht.

```
Begin transaction;
select *, IsValidReason(geometry), st_AsText(geometry) from line_layer_3d
where isvalid(geometry) <> 1;
Delete from line_layer_3d
where isvalid(geometry) <> 1;
Commit;
```

In der Tabelle *polyg_layer_3d* sind zwei sich selbst überschneidende Gemoetrien. Die visuelle Prüfung zeigt erst kein offensichtliches Problem, in einem Fall ist bei sehr kleinem Maßstab eine Kreuzung der Kontur durch sehr dicht gesetzte Punkte zu erkennen. Bei der Einmessung wurde das Prisma entweder zu stark geneigt, wodurch der Messpunkt hinter den vorangehenden berechnet wurde oder aber der Endpunkt der Linie wurde über versehentlich den Startpunkt hinaus gemessen. Beides kann leicht passieren und ist hier sicher nicht die Ausnahme.

Eine schnelle, zielorientierte Lösung scheint angebracht. Zuerst wird die Funktion st_makevalid() angewendet. Diese erstellt bei Überschneidungen aber ein Multipolygon und damit hier nicht zulässige Geometrien die auszuschließen sind.

Das Polygon wird mit der Funktion ST_SimplifyPreserveTopology() unter Wahrung der Topologie und einer relativ geringen Toleranz von 0.05 m vereinfacht. Da hierbei aber alle Punkte des Polygons berücksichtigt werden weicht die Fläche doch erkennbar vom Original ab und es werden zudem mit der

Funktion ST_Snap() die Knoten und die Kanten erneut an das original mit der selben Toleranz gefangen. Im Ergebnis liegt ein valides z-Polygon sehr großer Ähnlichkeit vor.

```
Begin transaction;
-- Probleme lokalisieren
select *, IsValidReason(geometry) from polyg_layer_3d
  where IsValid(geometry) <> 1;
-- Die erste einfache Problemlösung
Update polyg_layer_3d
  set geometry = ST_MakeValid(geometry)
  where IsValid(geometry) <> 1 and CastToSingle(ST_MakeValid(geometry)) not null;
-- Die zweite einfache Problemlösung im Vergleich
select st_area(ST_SimplifyPreserveTopology(geometry,0.05)) as simple_area,
 st_area(geometry) as orig_area,
 st_area(st_snap(ST_SimplifyPreserveTopology(geometry,0.05),geometry,0.05))
   as snap_simple_area,
 geometrytype(st_snap(ST_SimplifyPreserveTopology(geometry,0.05), geometry,0.05))
   as snap_simple_type,
 st_isvalid(st_snap(ST_SimplifyPreserveTopology(geometry,0.05), geometry,0.05))
   as snap_simple_valid
from polyg_layer_3d
where isvalid(geometry) <> 1;
Commit;
```

simple_area	orig_area	snap_simple_area	snap_simple_type	snap_simple_valid
1.699869	1.729045	1.729621	POLYGON Z	1

Das verbleibende ungültige Polygon wird mit dieser letzten Methode repariert.

```
Update polyg_layer_3d
  set geometry = st_snap(ST_SimplifyPreserveTopology(geometry,0.05),geometry,0.05)
  where isvalid(geometry) <> 1;
```

Alternativ könnten die aus ST_MakeValid() resultierenden Multipolygone aufgelöst, verglichen als auch bewertet und resultierend nur das repräsentative zum Original aufgehoben werden. Der Aufwand ist hier deutlich größer. Für die hier aufgezeigte Problemlösung muss aber: - das eingangs dargestellte Problem kleiner Überschneidungen vorliegen und - eine dem Maßstab als auch der Dimension angemessene Toleranz gewählt werden.

2.2.3 Höhenfehler korrigieren

Problem: Einzelne Punkte von Linien oder Polygone wurden an anderen Objekten gefangen oder von Hand gezeichnet, damit haben diese erkennbar einen falschen Z-Wert. Zum Beispiel:

- Die als Blöcke mit Attribut um die nördliche Grabungsfläche auf der Obrfläche eingemessenen Niv-Werte liegen bei 52,3 m im Westen und zwischen 53,0 und 53,4 m im Norden und Osten. Befundgrenzen sollten demnach mindestens 0,3 m unter diesen Werten liegen.
- Im Norden grenzen einige Befunde an die Schnittgrenze und die Befundlinien des Planum 1 fangen die Schnittgrenze auf dem Planum 0.
- Die Ansichtshaken der Profile sind von Hand gezeichnet und fallen damit auf die Höhe Null.
- Einige Befunde scheinen von Hand entlang gemessene Befundgrenzen konstruiert worden zu sein wobei die ohne Objektfang erstellen Knoten die Höhe 0 haben.

Daneben kann auch das grundsätzliche Problem einer falschen Prismenhöhe vorliegen. Daran schließt sich die Frage nach dem ursprünglich, richtigen z-Wert an, die aber nur näherungsweise beantwortet werden kann. Mögliche, annähernd korrekte Werte sind die nicht bei 0 liegenden z-Werte vor und hinter dem Knoten, der Durchschnitt oder Median der z-Wert des Polygons ohne die eindeutig “falschen” Werte oder der z-Wert des nächsten gemessenen Höhenwertes oder oder oder. Dazu kommt noch das Problem großen Polygonen wir der Grabungsfläche, die bei größzügiger Vermessung durch aus einen deutlichen Höhenunterschied von Punkt zu Punkt aufweisen können.

2.2.3.1 Tabelle Blöcke mit Attribut

Importieren wir deshalb als erstes die in AutoCAD exportierten Blöcke mit Attribut (Datenextraktion). Über das Menü "Menu > Advanced > Load CSV/TXT" rufen Sie das Importfenster auf, wählen die Datei "JUff_blockattr.csv", Table name: block_layer_3d, [x] First line contains ..., Text separator: [o] none, Column seperaor: [o] Comma, Decimal separator: Point und Charset Encoding: UTF-8 (ganz am Ende). Die Spalte sind Namen (Blockname), HÖHE (Attribut), Layer (Layer des Blocks), Position_X/Y/Z (Koordinaten des Blocks) und Nummer (Attribut). Nachfolgend werden zuerst eine Geometrie aus den Koordinaten und dann ein Überblick über die vorhandenen z-Werte erstellt.

```
Begin transaction;
Select AddGeometryColumn('block_layer_3d', 'geometry', 31468, 'POINTZ', 'XYZ');
Update block_layer_3d
  set geometry = ST_GeomFromText('POINTZ(' || "Position_X" || ' ' ||
  "Position_Y" || ' ' || "Position_Z" || ')', 31468);
select max(HÖHE) as max_H, min(HÖHE) as min_H, max(Position_Z) as max_Z,
  sum(Position_Z) / count(Position_Z) as mean_Z, min(Position_Z) as min_Z
from block_layer_3d group by name;
Commit;
```

Name	max_H	min_H	max_Z	mean_Z	min_Z
HP_FERTIG_OBERKANTE	53.690	50.540	53.832	51.724	0.126
HP_OHNE_SYMBOL	51.450	51.450	51.590	51.590	51.590
MESSPUNKT			52.920	52.507	50.977

Insgesamt scheinen die Messwerte plausibel zwischen 51 m und 54 m über NN zu liegen, doch auch hier gibt es bei den gemessenen z-Werten offensichtlich falsche Werte nahe 0.

```
select max(st_maxz(geometry)) as max_maxz,
max(st_minz(geometry)) as max_minz,
sum(st_maxz(geometry) - st_minz(geometry)) / count (*) as mean_diff,
min(st_maxz(geometry) - st_minz(geometry)) as min_diff,
max(st_maxz(geometry) - st_minz(geometry)) as max_diff
from line_layer_3d;
```

max_maxz	max_minz	mean_diff	min_diff	max_diff
105.604000	105.604000	3.323629	0.000000	52.732000

Die selbe Abfrage, nur für die Tabelle polyg_layer_3d.

max_maxz	max_minz	mean_diff	min_diff	max_diff
53.757953	53.026502	0.703233	0.000000	51.742000

Beginnen wir mit der Tabelle **block_layer_3d** und vertrauen wir darauf, dass die dargestellten, sichtbaren Höhenwerte visuell geprüft wurden und damit korrekt sein sollten. Wir verschaffen uns einen Überblick, indem wir die Differenz zwischen dargestelltem und gemessenem Wert (HÖHE - Position_Z) in Klassen von 1 cm zusammenfassen.

```
select count(*) as n, round((HÖHE - Position_Z)*100, 0) as diff_cm
from block_layer_3d
where abs(HÖHE - Position_Z) > 0.01
group by round((HÖHE - Position_Z)*100, 0);
```

diff_cm	-152	-114	-21	-18	-17	-16	-15	-14	-13	-9	22	5157
n	1	1	1	1	1	1	2	283	7	1	1	1

Das transponierte Ergebnis zeigt sehr zahlreiche Abweichungen um -14 cm und nur vereinzelt über +/- 15 cm hinaus. Die Häufung bei -14 cm weist auf ein strukturelles Problem, für das aber keine Lösung unmittelbar auf der Hand liegt. Auch wenn die Differenz bis 15 cm eigentlich nicht vorhanden sein sollte wird diese nachfolgend ignoriert und nur die um mehr als +/- 14 cm abweichenden Messwerte mit den Angaben aus HÖHE korrigiert

```
Update block_layer_3d
  set geometry = ST_GeomFromText('POINTZ(' || "Position_X" || ' ' ||
  "Position_Y" || ' ' || "HÖHE" || ')', 31468)
  where abs(HÖHE - Position_Z) > 0.14;
```

Die z-Werte aller Punkte liegen nun in einem plausiblen Wertebereich: Min.: 50.677, Max.: 53.829, Mittelw.: 51.981.

```
select min(ST_Z(geometry)) as min_z, max(ST_Z(geometry)) as max_z,
  sum(ST_Z(geometry))/count(*) as mean_z
  from block_layer_3d;
```

2.2.3.2 Tabelle line_layer_3d

Wie bei den Punkten ist die grundlegende Frage: Was erachten wir als falsch und wollen es korrigieren? Zwei Parameter lassen sich leicht berechnen: die Differenz innerhalb der Linie und der Quotient aus der Differenz und der Länge der Linie.

```
select feature_id, layer, st_maxz(geometry) - st_minz(geometry) as d,
    st_length(geometry) / (st_maxz(geometry) - st_minz(geometry)) as l_d,
    st_minz(geometry) as min_z, st_maxz(geometry) as max_z,
    st_npoints(geometry) as n_points, st_length(geometry) as length
from line_layer_3d
where d > 0.3 or l_d < 15
order by l_d desc;
```

Am Anfang der Liste stehen lange Linien bei denen ein Höhenunterschied zwischen 0.3 und 0.6 m durchaus plausibel ist. So ab einem Wert von kleiner als 15 für den Quotienten aus Länge zu Höhendifferenz wird es fraglich (feature_id 52) und bei dem Quotienten unter 13 (feature_id 14) vermutlich falsch. Für die Bearbeitung ergeben sich zwei Optionen: 1. Sie Schreiben die feature_id's der Linien in einen Filter und bearbeiten jede Linie in QGIS von Hand. 2. Sie verändern die Werte halbautomatisch und kontrollieren.

```
select group_concat(feature_id)
from line_layer_3d
where st_maxz(geometry) - st_minz(geometry) > 0.3 or
 st_length(geometry) / (st_maxz(geometry) - st_minz(geometry)) < 15;
```

Zuerst der Weg der reinen Handarbeit: Kopieren Sie zuerst die Liste der feature_id's der vorangehenden Abfrage in ein Protokoll Ihrer Änderungen am Datenbestand. Übertragen Sie die Filterbedingung nach `where` mit *copy 'n paste* in [Abfrage erstellen] unter dem Register Quelle bei den Eigenschaften des Layers in QGIS. Danach sehen Sie nur noch die potentiell falschen Linien und können diese editieren. Wählen Sie im Editiermodus das Stützpunktwerkzeug und markieren Sie mit einem rechten Mausklick eine Linie. Danach sollte sich ein Fenster mit der tabellarischen Liste der Koordinaten für jeden Knoten dieser Linie öffnen wo Sie dann die Wert verändern können.

Im vorliegenden Fall sind vor allem die vielen kurzen Profilinien die Masse des Problems. Sortiert nach dem kleinsten z-Wert fallen die ersten 55 Linien mit einem Wert weit unter 50 deutlich aus dem Rahmen des plausiblen. Ich verändere nachfolgend die Filterbedingung dahingehend, wobei aber mindestens ein plausibler z-Wert von über 50 m über NN vorliegen soll und schreibe alle betroffenen Linien in eine temporäre Tabelle. Je nach Anforderung können Sie ergänzend auch auf die Länge der Linie st_length(geometry), die Anzahl der Knoten st_npoints(geometry) oder auch weitere Eigenschaften filtern.

```
Begin transaction;
Drop table if exists line_err;
Create table line_err as
  select * from line_layer_3d
  where st_minz(geometry) < 50 and st_maxz(geometry) > 50;
Select RecoverGeometryColumn('line_err', 'geometry', 31468, 'linestringz', 'xyz');
Commit;
```

Danach werden die Linien in Multipoints aufgelöst und diese wiederum in eine Tabelle mit fortlaufenden Koordinaten. Für den letzten Schritt wird ein join zur Tabelle ElementaryGeometries erstellt. Hierbei werden jedes mal die Geometrien in der datenbank registriert und abschließend eine Tabelle zur visuellen Kontrolle erstellt.

```
Begin transaction;
-- Create table with multipoints per line
Drop table if exists line_err_multipoints;
Create table line_err_multipoints as
  select feature_id as line_feature_id, layer, st_dissolvepoints(geometry) as geometry
  from line_err;
Select RecoverGeometryColumn('line_err_multipoints', 'geometry', 31468,
   'multipointz', 'xyz');
-- Create table of individual points with a join  to ElementaryGeometries
Drop table if exists line_err_points;
create table line_err_points as
SELECT lmp.line_feature_id, lmp.rowid, e.item_no, e.geometry,
  ST_x(e.geometry) as x, ST_y(e.geometry) as y, ST_Z(e.geometry) AS z
FROM line_err_multipoints AS lmp
JOIN ElementaryGeometries AS e ON (e.f_table_name = 'line_err_multipoints'
AND e.origin_rowid = lmp.rowid);
Select RecoverGeometryColumn('line_err_points', 'geometry', 31468, 'pointz', 'xyz');
select * from line_err_points;
commit;
```

Die Tabelle enthält die ursprungliche *feature_id* der Linie, die *rowid* des ursprünglichen Multipoints und einen bei 0 beginnenden Zähler für jeden Knoten einer Linie. Die nächsten Schritte sind von der Überlegung her einfach: Ersetze jeden nicht plausiblen z-Wert durch den Mittelwert der plausiblen z-Werte der zugehörigen Linie. Schreibe danach die Punkteometrie neu und aggregiere die Punkte für jede ehemaligen Linie wieder zu einer Linie. Vergessen Sie nicht ggf. die Tabelle der genuin falschen z-Werte oder mindestens die *feature_id* der editierten Linien in einem Protokoll zu speichern (s.o.).

```
Begin transaction;
Drop table if exists line_err_points_new;
-- Create a new table with new z-values
Create table line_err_points_new as
-- counter for rowid and point_item, mean z and new z-value, order of case matters
with recursive point (mean_z, rid, z_new, item) as (
-- only to initialise the counters
 select
  (select sum(z) / count(z) from line_err_points
    where rowid = 1 and z > 50 and z < 55) as mean_z,
  0 as rid,
  NULL as z_new,
  -1 as item
 union all
-- get the values or count + 1 if point_item does not exist
 select
  case when (select z from line_err_points
               where rowid = rid and item_no = item + 1) not null
   then mean_z
```

```
   else (select sum(z) / count(z) from line_err_points
           where rowid = rid + 1 and z > 50 and z < 55)
   end as mean_z,
  case when (select z from line_err_points
              where rowid = rid and item_no = item + 1) not null
   then rid
   else rid + 1
   end as rid,
  case when (select z from line_err_points
              where rowid = rid and item_no = item + 1) > 50
   then (select z from line_err_points where rowid = rid and item_no = item + 1)
   else mean_z
   end as z_new,
  case when (select z from line_err_points
              where rowid = rid and item_no = item + 1) not null
   then item + 1
   else -1
   end as item
 from point
-- set the limit for the recursion
  where rid <= (select max(rowid) from line_err_points))
-- do something
Select a.rid, b.line_feature_id, b.item_no, a.mean_z, a.z_new, b.z, b.x, b.y
 from point as a
 inner join line_err_points as b on a.rid = b.rowid and a.item = b.item_no;
-- End of create table and now show the content
Select * from line_err_points_new;
Commit;
```

Kontrollieren Sie in der resultierenden Tabelle stichprobenartig zahlreiche z-Werte. Sollte das alles gut aussehen erstellen wir daraus eine neue Tabelle von Linien der ursprünglichen *feature_id*.

```
Begin transaction;
Drop table if exists line_err_new;
Create table line_err_new as
  Select line_feature_id as feature_id,
  MakeLine(MakePointZ(x, y, z_new, 31468)) as geometry
  from line_err_points_new
  group by line_feature_id;
Select RecoverGeometryColumn('line_err_new', 'geometry', 31468, 'linestringz', 'xyz');
Commit;
```

Zwischenergebnis nochmals kontrollieren und dann die alte Geometrie durch die veränderte Geometrie ersetzen.

```
Update line_layer_3d as a
set geometry = (select geometry from line_err_new as b
  where b.feature_id = a.feature_id)
where a.feature_id in (select feature_id from line_err_new);
```

Filtern Sie danach erneut die Linien mit weiterhin potentiell falschen Höhenwerten und nehmen sie die notwendigen Änderungen von Hand vor. Dies können Sie entweder wie oben dargestellt in QGIS tun (s.o.). Der wesentliche Vorteil, sie sehen die Linien im Kontext und können z.B. bewusst an die Anschlussinie fangen. Oder aber Sie generieren erneut eine Tabelle mit ausgelesenen Korrdinaten je Knoten, editieren diese Tabelle von Hand anstelle der Rekursionsabfrage, archivieren diese Tabelle als Protokoll Ihrer Änderungen und generieren hieraus wie zuvor eine Tabelle neuer Linien. Noch 33 Linien haben eine internen Höhendifferenz von mehr als 30 cm. Nach der Visualisierung in QGIS sind es u.a. einige Befunde an der nördlichen Grabungsgrenze und die Linien des neolithischen Grabensystems.

```
select group_concat(feature_id)
from line_layer_3d
where st_maxz(geometry) - st_minz(geometry) > 0.3;
```

2.2.3.3 Tabelle point_layer_3d

Die Daten scheinen insgesamt plausibel.

```
select group_concat(feature_id)
from line_layer_3d
where st_maxz(geometry) - st_minz(geometry) > 0.3;
```

layer	n	min_z	max_z
Höhenpunkte Befunde	274	51.975	52.928
Keramik Pl.1	72	52.211	52.905
Knochenreste	2	52.344	52.505
Photogrammetriepunkte	15	50.977	52.608
Profilzeichennägel	5	52.033	52.368

2.2.3.4 Tabelle polyg_layer_3d

Der Prozess der Höhenkorrektur kann analog zu den Tabelle der Linien durchgeführt werden, wobei die Schritte leicht angepasst werden müssen. Bei der folgenden Abfrage wird nicht die Länge der Linie (*st_length()*) sondern der Umfang verwendet (*st_perimeter()*). Die Abfrage mit den gewählten Parametern, Höhendifferenz über 0,1 m oder weniger als der 30-fache Umfang der Höhendifferenz, ergeben nur 12 potentiell fehlerhafte Befunde. In diesem Fall übertrage ich die Filteranweisung in eine Abfrage der Datenquele bei QGIS. Vor allem, um die ungewöhnliche Höhe von über 53 m über NN im räumlichen Kontext zu kontrollieren.

```
select feature_id, layer, st_maxz(geometry) - st_minz(geometry) as d,
    st_perimeter(geometry) / (st_maxz(geometry) - st_minz(geometry)) as p_d,
    st_minz(geometry) as min_z, st_maxz(geometry) as max_z,
    st_npoints(geometry) as n_points, st_perimeter(geometry) as p
from polyg_layer_3d
where d > 0.10 and p_d < 30
order by p_d desc;
```

feature_id	layer	d	p_d	min_z	max_z	n_points	p
348	Befunde	0.112	26.366	51.888	52.000	14	2.953
380	Befunde	1.411	21.489	51.890	53.301	114	30.322
379	Befunde	1.381	3.940	51.920	53.301	25	5.441
378	Befunde	1.361	2.719	51.940	53.301	16	3.700
376	Befunde	1.369	1.711	51.940	53.309	15	2.343
377	Befunde	1.350	0.882	51.960	53.310	10	1.190
317	Befunde	51.670	0.422	0.140	51.810	77	21.848
294	Befunde	51.571	0.159	0.140	51.711	27	8.211
326	Befunde	51.742	0.102	0.140	51.882	20	5.296
425	Befunde	51.567	0.076	0.140	51.707	16	3.955
424	Befunde	51.201	0.072	0.140	51.341	13	3.686
476	Befunde	51.227	0.056	0.140	51.367	16	2.910

Der erste Befund (*feature_id* 348) zeigt bei der Autopsie keine Auffälligkeit. Der nächste Befund (*feature_id* 380) hat im Vergleich mit den benachbarten Befunden bei Start- und Endpunkt einen plausiblen Höhenwert, die anderen Knoten liegen aber ca. 1,4 m höher. Der selbe Höhenfehler betrifft auch die folgenden vier Befunde aus der unmittelbaren Nachbarschaft. Hier scheint ein typischer Fehler einer falschen Prismahöhe vorzuliegen. Da insgesamt 175 z-Werte geändert werden müssten verschiebe ich diese Befunde als ganzes um 1,4 nach unten und ändere nur den jeweiligen Start- und Endpunkt von Hand wieder auf die ursprüngliche Höhe.

```
Update polyg_layer_3d
set geometry = ST_Translate(geometry, 0, 0, -1.4)
where feature_id in (376, 377, 378, 379, 380);
```

2.2.3.5 Tabelle text_layer_3d

Eine Tabelle zu den Höhenwerten offenbart wirklich seltsame z-Werte bei dem ehemaligen Absatztext (M-Text) auf den Layern '00BefundNr' und 'Kabelschacht'.

select layer, count(*) as n, min(ST_Z(geometry)) as min_z, max(ST_Z(geometry)) as max_z from text_layer_3d group by layer;

layer	n	min_z	max_z
00BefundNr	262	2950.925	3250.135
00ProfilNr	215	0.140	51.373
Befundnummern	279	0.140	105.955
Flächengrenzen	2	0.140	0.140
Kabelschacht	1	3108.713	3108.713
Profile	1	0.140	0.140

Der Import von Text in Spatialite ist Ansich sehr gut und das hier vorliegenede Problem liegt genuin in der AutoCAD-Datei und dem ursprünglichen Absatztext. Normalerweise wird eine Textbox (2D) von einem Punkt aus aufgezogen und der zugehörige Basispunkt ist *a priori* die obere linke Ecke. Wird kein z-Wert übergeben liegt die Textbox auf 0. Wird der Text aufgelöst bekommt jede resultierende Einzelzeile den linken Punkt (Startpunkt) der Grundlinie als Einfügekoordinate (Geometrie-Position: x, y z) mit der Textausrichtung: 0, 0, 0. Wird der Text dann anders ausgerichtet, z.B. rechtsbündig, wird die Einfügekoordiate (Geometrie-Position angepasst und die Koordiaten der alten Geometrie-Position werden unter der Textausrichtung gespeichert. In der vorliegenden AutoCAD Datei ist dies Beziehung zwischen M-Text, Text und der Textausrichtung nicht nachvollziehbar verändert.

Zwei Optionen in solchen Fällen: 1. Die z-Werte interessieren Sie nicht, weil die räumliche Verknüpfung zwischen Befundnummer und Polygon zweidimensional erfolgt und das Polygon ja eine korrekte Höhe aufweist. 2. Sie exportieren in AutoCAD alle Textzeilen mit beiden Koordinatensystemen analog zu den Blöcken mit Attribut unter verwendung von "Extras > Datenextraktion" in eine CSV-Datei.

Die z-Werte der Texte auf dem Layer 'Befundnummern' sind aUch nicht wirklich vertrauenerweckend, 222 Nummern liegen auf 0.14, 37 auf 53.221 und 11 auf 105.955 m über NN.

2.2.3.6 Aufräumen und Konsistenz Prüfung

Wegen der vielen Änderungen aktualisiere ich die interne Statistik und prüfe auch die Geometrien der Tabellen erneut.

```
Begin transaction;
Select UpdateLayerStatistics('block_layer_3d', 'geometry')
Union all
Select UpdateLayerStatistics('line_layer_3d', 'geometry')
Union all
Select UpdateLayerStatistics('point_layer_3d', 'geometry')
Union all
Select UpdateLayerStatistics('polyg_layer_3d', 'geometry')
Union all
Select UpdateLayerStatistics('text_layer_3d', 'geometry');
-- chech geometries
Select 'block' as tbl, IsValid(Geometry), count(*)
  from block_layer_3d group by IsValid(Geometry)
Union all
Select 'line', IsValid(Geometry), count(*) from line_layer_3d
  group by IsValid(Geometry)
Union all
```

```
Select 'point', IsValid(Geometry), count(*) from point_layer_3d
  group by IsValid(Geometry)
Union all
Select 'polyg', IsValid(Geometry), count(*) from polyg_layer_3d
  group by IsValid(Geometry)
Union all
Select 'text', IsValid(Geometry), count(*) from text_layer_3d
  group by IsValid(Geometry);
Commit;
```

2.2.4 Datenbearbeitung

Datenaufbereitung erfolgt analog der Darstellung zur Dokumentation des Fundplatzes Wittelsberg Fpl. 7. Im Unterschied hierzu muss wegen der heterogenen Layernamen deutlich mehr Informationsbereinigung betrieben werden. Die Struktur der Geometrietabellen wird entsprechend *a priori* um die Spalten 'number (integer)' und 'info (text)' erweitert.

2.2.4.1 Tabelle line_layer_3d

Ergänzen wir die Spalten und schaffen wir uns erstmal einen Überblick über die Layer und den jeweils vorhandenen Linien.

```
Begin transaction;
ALTER TABLE "line_layer_3d" ADD COLUMN number integer;
ALTER TABLE "line_layer_3d" ADD COLUMN info text;

select group_concat(l, ', ')
from (select layer || ': ' || count(*) as l, 'g' as g
  from line_layer_3d
  group by layer)
group by g;
Commit;
```

00Befund: 18, 00Profil: 537, Befunde: 77, Befunde Planum 2: 3, Befundnummern: 9, GUK: 16, Keramik Pl.1: 1, Pl1-Keramik: 6, Pl2-Befunde: 3, Pl2-Keramik: 2, Pl2-Knochen: 2, Profile: 198, Umrandung Hausgrundriss: 14

'Profile' befinden sich im Norden, '00Profile' im Süden der Ausgrabung. Beides kann problemlos unter Profile geführt werden. Das Gleiche gilt für 'Befunde' und '00Befund'. Der Layer 'Befundnummern' zeigt zahlreiche Linien aus zwei Punkten bei denen es sich um Verweislinien für weit ausgestellte Befundnummern handelt. 'Befunde Planum 2' können 'Pl2-Befunde' zugewiesen werden. Die insgesamt 11 Linien der Fundlayer mit 'Keramik' oder 'Knochen' sind sehr kleine Kreise von max. 4 cm Durchmesser, diese werden mit dem jeweiligen Zentroid und dem z-Wert des (2D) Kreises an die Tabelle point_layer_3d angefügt. Die Linien von 'Umrandung Hausgrundriss' sind zahlreiche Pfosten, deren Deutung der Tabelle hatch_layer_2D zugewiesen wird. Hierbei muss die 3D-Linie zu einem 2D Multipolygon umgewandelt werden. Da die Befunde als Polygon bereits vorliegen können sie gelöscht werden.

```
Begin transaction;
Update line_layer_3d set layer = 'Profile' where layer = '00Profil';
Update line_layer_3d set layer = 'Befunde' where layer = '00Befund';
Update line_layer_3d set layer = 'Pl2-Befunde' where layer = 'Befunde Planum 2';
insert into point_layer_3d (filename, layer, geometry)
  select filename, layer,
  MakePointZ(ST_X(ST_Centroid(geometry)), ST_Y(ST_Centroid(geometry)),
     ST_MaxZ(geometry), 31468)
  from line_layer_3d where layer like '%Keramik%' or layer like '%Knochen%';
Delete from line_layer_3d
  where layer like '%Keramik%' or layer like '%Knochen%';
Insert into hatch_layer_2d (filename, layer, geometry)
  select filename, layer, CastToMulti(CastToXY(ST_MakePolygon(geometry))) as geometry
```

```
  from line_layer_3d
  where layer = 'Umrandung Hausgrundriss';
Delete from line_layer_3d
  where layer = 'Umrandung Hausgrundriss';
Commit;
```

Die Linien der Layer Befunde und GUK (Grabungsgrenze Unkterkante?) müssen in Polygone konvertiert und in die Tabelle polyg_layer_3D übertragen werden. Beginne wir mit 'GUK', hier liegen 16 Einzellinien vor, die in zwei Polygone aggregiert werden müssen, deshalb ST_Polygonize(). Es resultiert ein Multipolygon mit zwei Polygonen. Der Pragmatische Weg ist diese gezielt abzufragen und anzüfugen.

```
Begin transaction;
select GeometryType(ST_Polygonize(geometry)) as typ,
  NumGeometries(ST_Polygonize(geometry)) as n
  from line_layer_3d
  where layer = 'GUK';
Insert into polyg_layer_3d (filename, layer, geometry)
  select filename, layer, GeometryN(ST_Polygonize(geometry), 1)
  from line_layer_3d where layer = 'GUK';
Insert into polyg_layer_3d (filename, layer, geometry)
  select filename, layer, GeometryN(ST_Polygonize(geometry), 2)
  from line_layer_3d where layer = 'GUK';
Delete from line_layer_3d where layer = 'GUK';
Commit;
```

Zahlreiche weitere Befunde liegen aus diversen Gründen nur als Linien vor. Einige davon könnten wir als Polygone aus hatch_layer_2d kopieren, dann aber ohne einen Höhenwert.Inm vorliegenden Fall sind die Linien im GIS auch leicht verschoben, was mir auch schon bei den Polygonen aus der Schraffur aufgefallen war. Ich vermute hier Projektionsproblem. Beginnen wir mit dem offensichtlich einfachsten Problem, den Kreisen, bzw. allen geschlossenen Polygonen, die auf Anhieb ein Polygon ergeben sollten. Dies sind 27, deren feature_id für eine visuelle Kontrolle in QGIS mit einer Abfrage 'layer in (1, 2, .., n)' direkt in einen Vektor geschrieben werden. Es folgt die Anfügeabfrage und die Löschabfrage für die erfolgreich konvertierten Linien. Die Kreise haben leider einen erkennbar falschen Höhenwert, da die Erhebung aus AutoCAD nicht übernommen wurde. Dies wird später anhand der Befundnummern aus block_layer_3d korrigiert.

```
Begin transaction;
-- Was kann direkt ein Polygon werden?
select group_concat(feature_id, ', ') from line_layer_3d
where isvalid(ST_BuildArea(geometry)) = 1
-- Polygone anfügen
Insert into polyg_layer_3d (filename, layer, geometry)
 select filename, layer, ST_BuildArea(geometry) as geometry from line_layer_3d
 where layer like '%Befund%' and isvalid(ST_BuildArea(geometry)) = 1;
-- Zugehörige Linien löschen
Delete from line_layer_3d
 where layer like '%Befund%' and isvalid(ST_BuildArea(geometry)) = 1;
Commit;
```

Es verbleiben etliche Befunde, die entlang einer Grabungsgrenze, an am Profil auf Planum 2 oder als Konstruierte Befunde im Planum geschlossen werden können. Dies können wir erstmal pauschal angehen und bei Bedarf im Nachgang für einen exakten Verlauf an der Grabungsgrenze oder dem Profil Knoten von Hand ergänzen (z.B. *feature_id* 779). Das Problem löse ich "von Hand" durch eine Liste der betroffenen *feature_id*'s.

```
Begin transaction;
Insert into polyg_layer_3d (filename, layer, geometry)
  select filename, layer,
     ST_BuildArea(ST_AddPoint(geometry, ST_StartPoint(geometry))) as geometry
  from line_layer_3d
```

```
  where feature_id in (2, 3, 7, 8, 10, 11, 15, 559, 560, 561, 777, 778, 779, 786, 789,
  790, 791, 792, 793, 794, 800, 801, 802, 803);
Delete from line_layer_3d
  where feature_id in (2, 3, 7, 8, 10, 11, 15, 559, 560, 561, 777, 778, 779, 786, 789,
  790, 791, 792, 793, 794, 800, 801, 802, 803);
Commit;
```

Nach er visuellen Kontrolle der verbleibenden Linien liegen noch drei Befunde aus aus einzelnen Linien vor und die beiden langen Gräben. Für alle Befunde wird abschließend mit *st_polygonize()* ein Polygon gebildet. Da diese Funktion aggregiert alles, kann also auch für mehr als zwei Linien eingesetzt werden, schreibt aber im ersten Fall die drei erwünschten Polygone in ein Multipolygon. Anstatt nachträglich zu trennen gehen wir das individuell an.

```
Begin transaction;
Insert into polyg_layer_3d (filename, layer, geometry)
  select filename, layer, st_geometryN(st_polygonize(geometry),1)
  from line_layer_3d
  where feature_id in (798,799);
Insert into polyg_layer_3d (filename, layer, geometry)
  select filename, layer, st_geometryN(st_polygonize(geometry),1)
  from line_layer_3d
  where feature_id in (788, 796);
Insert into polyg_layer_3d (filename, layer, geometry)
  select filename, layer, st_geometryN(st_polygonize(geometry),1)
  from line_layer_3d
  where feature_id in (795, 787);
Delete from line_layer_3d
  where feature_id in (798,799, 788, 796, 795, 787);
Commit;
```

Bei den beiden Gräben, bzw. den drei dokumentierten Grabenabschnitten müssen die fehlenden Linien ergänzt werden. Danach ist die Kombination der Teillinien mit *ST_Polygonize()* zu einem Multipolygon und das Anfügen der Teilpolygone möglich.

*Wichtig**: Die Felder ‘filename’ und ‘layer’ dürfen nicht leer (*NULL*) sein dürfen.

```
Begin transaction;
select filename, layer, st_geometryN(st_polygonize(geometry),1)
  from line_layer_3d
  where feature_id in (4, 5, 13, 14, 16, 17, 18, 19, 784, 785, 891, 892, 893, 894, 895, 896);
select filename, layer, st_geometryN(st_polygonize(geometry),2)
  from line_layer_3d
  where feature_id in (4, 5, 13, 14, 16, 17, 18, 19, 784, 785, 891, 892,
  893, 894, 895, 896);
select filename, layer, st_geometryN(st_polygonize(geometry),3)
  from line_layer_3d
  where feature_id in (4, 5, 13, 14, 16, 17, 18, 19, 784, 785, 891, 892,
  893, 894, 895, 896);
Delete from line_layer_3d
  where feature_id in (4, 5, 13, 14, 16, 17, 18, 19, 784, 785, 891, 892,
  893, 894, 895, 896);
Commit;
```

Es verbleiben wenige Linien die nach eingehender Prüfung keinen Sinn ergeben und gelöscht werden. Zudem werden die Befunde hier nicht nach den Teillinien getrennt, dies muss bei Bedarf in Handarbeit erfolgen.

```
Delete from line_layer_3d
  where feature_id in (825, 824, 699, 822)
```

2.2.4.2 Tabelle point_layer_3d

Ergänzen wir die Spalten 'number' als auch 'info' und schaffen wir uns nochmals einen Überblick über die Layer und die jeweilige Anzahl an Punkten als Text für das Protokoll.

```
Begin transaction;
ALTER TABLE "point_layer_3d" ADD COLUMN number integer;
ALTER TABLE "point_layer_3d" ADD COLUMN info text;

select group_concat(l, ', ')
from (select layer || ': ' || count(*) as l, 'g' as g
  from point_layer_3d group by layer)
group by g;
Commit;
```

Höhenpunkte Befunde: 274, Keramik Pl.1: 73, Knochenreste: 2, Photogrammetriepunkte: 15, Pl1-Keramik: 6, Pl2-Keramik: 2, Pl2-Knochen: 2, Profilzeichennägel: 5. Die Begriffe sollten einheitlicher und gestaltet werden.

Der neue Layername soll analog den Profilen und Befunden als Präfix das Planum erhalten, nachfolgend die Bezeichnung 'Funde' oder 'Messpunkt'. In die Spalte 'info' kommt die Objektbeschreibung und in der Spalte 'number' wird später die Befundnummer durch eine räumliche Verbindung eingetragen.

```
Begin transaction;
Update point_layer_3d
  set info = 'Keramik' where layer like '%Keramik%';
Update point_layer_3d
  set info = 'Knochen' where layer like '%Knochen%';
Update point_layer_3d
  set info = 'Photogrammetrie' where layer like '%Photogrammetrie%';
Update point_layer_3d
  set info = 'Profilnagel' where layer like '%Profil%';
Update point_layer_3d
  set info = 'Niv-Punkt' where layer like '%Höhen%';
Update point_layer_3d
  set layer = 'Pl02_Funde' where layer like '%Pl2-%';
Update point_layer_3d
  set layer = 'Pl01_Funde' where layer like '%Pl1-%' or layer like '%Pl.1';
Update point_layer_3d
  set layer = 'PL___Funde' where layer like '%Knochen%';
Update point_layer_3d
  set layer = 'PL___Messpunkt' where substr(layer,1,2) <> 'Pl';
Commit;
```

2.2.4.3 Tabelle polyg_layer_3d

Ergänzen wir die Spalten 'number' als auch 'info' und schaffen wir uns nochmals einen Überblick über die Layer und die jeweilige Anzahl an Objekten als Text für das Protokoll.

```
Begin transaction;
ALTER TABLE "polyg_layer_3d" ADD COLUMN number integer;
ALTER TABLE "polyg_layer_3d" ADD COLUMN info text;

select group_concat(l, ', ')
from (select layer || ': ' || count(*) as l, 'g' as g
  from polyg_layer_3d group by layer)
group by g;
Commit;
```

00Befund: 250, Befunde: 277, Flächengrenzen: 3, GGok: 1, GGuk: 2, GUK: 2, Grabungsgrenze: 1, Keramik Pl.1: 1, Pl2-Befunde: 6, Profile: 1

```
Begin transaction;

Commit;
```

```
Begin transaction;

Commit;
```

```
Begin transaction;

Commit;
```

3 Literatur