Introduction to SQL
with examples from digital soil survey data

D. E. Beaudette

Dept. Land, Air and Water Resources
University of California
Davis, California

debaudette@ucdavis.edu

October 13, 2008
**SQL**

1. "Structured Query Language"

### History
- developed by IBM in the '70s
- interactive vocabulary for database queries
- most modern systems are built on the 'SQL-92' standard

### Modern Uses
- general purpose question *asking* vehicle
- SQL-based interfaces to many types of data: filesystem elements, GIS data, etc.
- often abstracted behind an interface of some kind: i.e. Google, etc.

---

Flavors of SQL / Portability Issues

Many Vendors / Projects
- client/server: Oracle, MS SQL, Informix, IBM, MySQL, PostGreSQL
- file-based: Access, SQLite, BerkeleyDB

...but all support a subset of the SQL standards

Backwards Compatibility = Not Portable
- standard is vague on actual syntax
- complex & large standard → only subset implemented
- historic deviations from standard preserved

...in most cases the differences are slight
### SQL Extensions

#### Why Bother?
The SQL language is great for simple set operations, but lacks many of the convenient functions found in other languages. Extensions provide the ability to "call" external functions from other common programming languages, entirely within the context of the database.

#### Some Examples
- "procedural SQL": PL/SQL, SQL PL, PGPLSQL, etc.
- SQL/XML: parsing of XML (extensible markup language documents)
- SQL/R: use of R language commands (numerical algorithms, statistics, etc.)
- SQL/Perl: use of perl language commands libraries (pattern matching, etc.)
- SQL/Python: use of python language commands and libraries
SQL "Structured Query Language"

Syntax Notes

- set-based, declarative computer language
  i.e. a program that describes what to do, not how to do it

- 3-value logic: TRUE, FALSE, NULL

- several language elements:
  - statements: SQL code that has a persistent effect on tables, etc.
  - queries: SQL code that returns data
  - expressions: operations on or tests against a column’s contents
  - clauses: logical 'chunks' of statements / queries

```
UPDATE country
SET population = population + 1
WHERE name = 'USA';
```

\(^2\)image c/o wikipedia
SQL Syntax

Syntax Notes

```
SELECT [ columns ]
[ FROM from_item ]
[ WHERE condition ]
[ GROUP BY expression ]
[ HAVING condition ]
[ { UNION | INTERSECT | EXCEPT } SELECT [...] ]
[ ORDER BY expression [ ASC | DESC ] ]
[ LIMIT count ]
```

Example Query

```
SELECT column_x, column_y, column_z
FROM table_x
WHERE column_x = 'something'
     -- optional
GROUP BY column_x
ORDER BY column_x ; -- semi-colon denotes end of SQL statement
```
SELECT Statements

Give me the horizon names and depths for soil id '467038:646635'

SELECT cokey, hzname, hzdept_r, hzdepb_r -- the column names
FROM chorizon -- the table name
WHERE cokey = '467038:646635' -- filtering condition
ORDER BY hzdept_r ASC; -- ordering of results

Query Result

<table>
<thead>
<tr>
<th>cokey</th>
<th>hzname</th>
<th>hzdept_r</th>
<th>hzdepb_r</th>
</tr>
</thead>
<tbody>
<tr>
<td>467038:646635</td>
<td>Ap</td>
<td>0</td>
<td>18</td>
</tr>
<tr>
<td>467038:646635</td>
<td>Bw</td>
<td>18</td>
<td>41</td>
</tr>
<tr>
<td>467038:646635</td>
<td>Bk1</td>
<td>41</td>
<td>69</td>
</tr>
<tr>
<td>467038:646635</td>
<td>Bk2</td>
<td>69</td>
<td>109</td>
</tr>
<tr>
<td>467038:646635</td>
<td>Bk3</td>
<td>109</td>
<td>145</td>
</tr>
<tr>
<td>467038:646635</td>
<td>Bk4</td>
<td>145</td>
<td>183</td>
</tr>
</tbody>
</table>
INSERT records into a table

**INSERT INTO** chorizon — table name
(cokey, hzname, hzdept_r, hzdepb_r) — record template
**VALUES** — SQL keyword 'here comes the data'
('new_cokey', 'Ap', 0, 10) — a new record

UPDATE existing records in a table

**UPDATE** chorizon — table to modify some records in
**SET** hzname = 'E' — update horizon names to modern conventions
**WHERE** hzname = 'A2' ; — but only the matching historic names

DELETE records FROM a table (be careful!)

**DELETE FROM** chorizon — table to delete records from
**WHERE** hzname IS NULL ; — records that are missing a horizon name
Table Modification Statements

### Altering Table Structure

— add a new column

```sql
ALTER TABLE chorizon ADD COLUMN hydrophobicity_index integer;
```

— now remove the column

```sql
ALTER TABLE chorizon DROP COLUMN hydrophobicity_index integer;
```

### Altering Column Definitions

— rename a column

```sql
ALTER TABLE chorizon RENAME COLUMN claytotal_r TO clay;
```

— change the column’s datatype (be careful!)

```sql
ALTER TABLE chorizon ALTER COLUMN clay TYPE numeric;
```

— do not allow NULL values in a column

```sql
ALTER TABLE chorizon ALTER COLUMN clay SET NOT NULL;
```

— do not allow values over 100%

```sql
ALTER TABLE chorizon ALTER COLUMN clay CHECK (clay <= 100);
```
Operations on a Single Table

- filtering by column: SELECT a, b, c, ...
- filtering by row: WHERE
- ordering: ORDER BY
- aggregating: GROUP BY
- aggregating *then* filtering: GROUP BY, HAVING
Filtering by Column

Return all columns, from all records from the chorizon table

```sql
SELECT * from chorizon;
```

Return the hzname column from the chorizon table, then rename it

```sql
SELECT hzname as horizon_name from chorizon;
```

Return then horizon top + bottom columns as a new column from the chorizon table

```sql
-- column concatenation: 'a' || 'b' --> 'ab'
SELECT hzdept_r || '-' || hzdepb_r as hz_interval from chorizon;

hz_interval
0–5
[...]
### Filtering by Row

*Return all horizons that have a clay content of $\geq 40\%$*

```sql
SELECT hzname, sandtotal_r, silttotal_r, claytotal_r
FROM chorizon
WHERE claytotal_r $\geq$ 40;
```

<table>
<thead>
<tr>
<th>hzname</th>
<th>sandtotal_r</th>
<th>silttotal_r</th>
<th>claytotal_r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bt2</td>
<td>26.1</td>
<td>28.9</td>
<td>45</td>
</tr>
<tr>
<td>Bt2</td>
<td>26.1</td>
<td>28.9</td>
<td>45</td>
</tr>
<tr>
<td>Bw</td>
<td>5.3</td>
<td>44.7</td>
<td>50</td>
</tr>
<tr>
<td>Az</td>
<td>7.2</td>
<td>47.8</td>
<td>45</td>
</tr>
<tr>
<td>Cz</td>
<td>26.1</td>
<td>28.9</td>
<td>45</td>
</tr>
<tr>
<td>Bw1</td>
<td>30</td>
<td>22.5</td>
<td>47.5</td>
</tr>
</tbody>
</table>

...note that the results aren't guaranteed to be returned in any specific order.

...an index will speed this operation up when you have lots of data.
Filtering by Row

Return a list of unique horizon names that match a given pattern

```
SELECT DISTINCT hzname
FROM chorizon
WHERE hzname LIKE '%A%'
AND areasymbol = 'ca653';
```

<table>
<thead>
<tr>
<th>hzname</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
</tr>
<tr>
<td>A1</td>
</tr>
<tr>
<td>A2</td>
</tr>
<tr>
<td>A3</td>
</tr>
<tr>
<td>AB</td>
</tr>
<tr>
<td>ABt</td>
</tr>
<tr>
<td>Ad</td>
</tr>
<tr>
<td>Agb</td>
</tr>
<tr>
<td>Anz</td>
</tr>
<tr>
<td>Ap</td>
</tr>
<tr>
<td>Ap1</td>
</tr>
<tr>
<td>[... ]</td>
</tr>
</tbody>
</table>
```
Filtering by Row

Return details from named map units

```
SELECT mukey, comppct_r, majcompflag, compname
FROM component
WHERE mukey IN ('459154', '459210', '459212', '459213')
AND majcompflag = 'Yes';
```

<table>
<thead>
<tr>
<th>mukey</th>
<th>comppct_r</th>
<th>majcompflag</th>
<th>compname</th>
</tr>
</thead>
<tbody>
<tr>
<td>459154</td>
<td>100</td>
<td>Yes</td>
<td>Water</td>
</tr>
<tr>
<td>459210</td>
<td>85</td>
<td>Yes</td>
<td>Balcom</td>
</tr>
<tr>
<td>459212</td>
<td>45</td>
<td>Yes</td>
<td>Balcom</td>
</tr>
<tr>
<td>459212</td>
<td>40</td>
<td>Yes</td>
<td>Dibble</td>
</tr>
<tr>
<td>459213</td>
<td>85</td>
<td>Yes</td>
<td>Brentwood</td>
</tr>
</tbody>
</table>

This is also possible

```
SELECT mukey, comppct_r, majcompflag, compname
FROM component
WHERE mukey IN (SELECT mukey FROM [...] )
AND majcompflag = 'Yes';
```
# Return some data ordered by multiple strata

**SELECT** [...]
**FROM** [...]
**ORDER BY** mukey **ASC**, cokey **ASC**, comppct_r **DESC**, hzdept_r **ASC**;

<table>
<thead>
<tr>
<th>mukey</th>
<th>cokey</th>
<th>comppct_r</th>
<th>top</th>
<th>bottom</th>
<th>db</th>
<th>om</th>
</tr>
</thead>
<tbody>
<tr>
<td>459210</td>
<td>459210:623942</td>
<td>85</td>
<td>0</td>
<td>61</td>
<td>1.6</td>
<td>0.75</td>
</tr>
<tr>
<td>459210</td>
<td>459210:623942</td>
<td>85</td>
<td>61</td>
<td>94</td>
<td>1.6</td>
<td>0.25</td>
</tr>
<tr>
<td>459210</td>
<td>459210:623942</td>
<td>85</td>
<td>94</td>
<td>104</td>
<td></td>
<td></td>
</tr>
<tr>
<td>459212</td>
<td>459212:623950</td>
<td>45</td>
<td>0</td>
<td>51</td>
<td>1.6</td>
<td>0.75</td>
</tr>
<tr>
<td>459212</td>
<td>459212:623950</td>
<td>45</td>
<td>51</td>
<td>76</td>
<td>1.6</td>
<td>2.5</td>
</tr>
<tr>
<td>459212</td>
<td>459212:623950</td>
<td>45</td>
<td>76</td>
<td>86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>459212</td>
<td>459212:623951</td>
<td>40</td>
<td>0</td>
<td>10</td>
<td>1.66</td>
<td>0.75</td>
</tr>
<tr>
<td>459212</td>
<td>459212:623951</td>
<td>40</td>
<td>10</td>
<td>76</td>
<td>1.68</td>
<td>0.25</td>
</tr>
<tr>
<td>459212</td>
<td>459212:623951</td>
<td>40</td>
<td>76</td>
<td>86</td>
<td></td>
<td></td>
</tr>
<tr>
<td>459213</td>
<td>459213:623953</td>
<td>85</td>
<td>0</td>
<td>25</td>
<td>1.81</td>
<td>0.75</td>
</tr>
<tr>
<td>459213</td>
<td>459213:623953</td>
<td>85</td>
<td>25</td>
<td>89</td>
<td>1.74</td>
<td>0.25</td>
</tr>
<tr>
<td>459213</td>
<td>459213:623953</td>
<td>85</td>
<td>89</td>
<td>152</td>
<td>1.78</td>
<td>0.25</td>
</tr>
</tbody>
</table>
Making a New Table from a SELECT Statement

Extract some data, renaming columns into a new table for later use

```sql
CREATE TEMP TABLE s_data as
SELECT cokey, hzname as name, hzdept_r as top, hzdepb_r as bottom, awc_r,
     (hzdepb_r - hzdept_r) * awc_r as awc_cm, claytotal_r as clay
FROM chorizon
WHERE areasymbol = 'ca653'
ORDER BY cokey, hzdept_r ASC;
```

Some of the data from the new table s_data

<table>
<thead>
<tr>
<th>cokey</th>
<th>name</th>
<th>top</th>
<th>bottom</th>
<th>awc_r</th>
<th>awc_cm</th>
<th>clay</th>
</tr>
</thead>
<tbody>
<tr>
<td>467013:646485</td>
<td>Ap</td>
<td>0</td>
<td>36</td>
<td>0.14</td>
<td>5.04</td>
<td>23</td>
</tr>
<tr>
<td>467013:646485</td>
<td>Bkg</td>
<td>36</td>
<td>56</td>
<td>0.14</td>
<td>2.8</td>
<td>25</td>
</tr>
<tr>
<td>467013:646485</td>
<td>Bkng</td>
<td>56</td>
<td>107</td>
<td>0.14</td>
<td>7.14</td>
<td>25</td>
</tr>
<tr>
<td>467013:646485</td>
<td>B'kg</td>
<td>107</td>
<td>152</td>
<td>0.14</td>
<td>6.3</td>
<td>25</td>
</tr>
<tr>
<td>467014:646490</td>
<td>Ap</td>
<td>0</td>
<td>74</td>
<td>0.16</td>
<td>11.84</td>
<td>22</td>
</tr>
<tr>
<td>467014:646490</td>
<td>Bg</td>
<td>74</td>
<td>87</td>
<td>0.14</td>
<td>1.82</td>
<td>7</td>
</tr>
<tr>
<td>467014:646490</td>
<td>Agb</td>
<td>87</td>
<td>98</td>
<td>0.15</td>
<td>1.65</td>
<td>18</td>
</tr>
<tr>
<td>467014:646490</td>
<td>B'g</td>
<td>98</td>
<td>111</td>
<td>0.14</td>
<td>1.82</td>
<td>8</td>
</tr>
<tr>
<td>467014:646490</td>
<td>A'gb</td>
<td>111</td>
<td>222</td>
<td>0.15</td>
<td>16.65</td>
<td>27</td>
</tr>
<tr>
<td>467015:646496</td>
<td>Ap1</td>
<td>0</td>
<td>33</td>
<td>0.16</td>
<td>5.28</td>
<td>30</td>
</tr>
<tr>
<td>467015:646496</td>
<td>Ap2</td>
<td>33</td>
<td>61</td>
<td>0.12</td>
<td>3.36</td>
<td>30</td>
</tr>
<tr>
<td>467015:646496</td>
<td>Bknzg</td>
<td>61</td>
<td>130</td>
<td>0.12</td>
<td>8.28</td>
<td>30</td>
</tr>
<tr>
<td>467015:646496</td>
<td>2Bknzg</td>
<td>130</td>
<td>183</td>
<td>0.1</td>
<td>5.3</td>
<td>20</td>
</tr>
</tbody>
</table>

[...]
Simple Aggregation

**Compute profile depth, sum AW,C and hz-thickness-weighted % clay**

```
SELECT cokey, sum(bottom - top) as soil_depth, 
sum((bottom - top) * awc_r) as profile_awc_cm, 
sum((bottom - top) * clay) / sum((bottom - top)) as wt_mean_clay 
FROM s_data 
GROUP BY cokey ;
```

**Results**

<table>
<thead>
<tr>
<th>cokey</th>
<th>soil_depth</th>
<th>profile_awc_cm</th>
<th>wt_mean_clay</th>
</tr>
</thead>
<tbody>
<tr>
<td>467104:647132</td>
<td>152</td>
<td>13.17</td>
<td>35.8684210526316</td>
</tr>
<tr>
<td>467166:659648</td>
<td>152</td>
<td>24.45</td>
<td>29.7236842105263</td>
</tr>
<tr>
<td>467074:646859</td>
<td>50</td>
<td>2.6</td>
<td>5.2</td>
</tr>
<tr>
<td>467079:646894</td>
<td>152</td>
<td>8.48</td>
<td>7.67105263157895</td>
</tr>
<tr>
<td>467033:646606</td>
<td>183</td>
<td>23.73</td>
<td>31.3551912568306</td>
</tr>
<tr>
<td>467159:659641</td>
<td>66</td>
<td>3.75</td>
<td>3.72727272727273</td>
</tr>
<tr>
<td>467024:646549</td>
<td>152</td>
<td>5.54</td>
<td>40.0065789473684</td>
</tr>
<tr>
<td>467078:646888</td>
<td>58</td>
<td>3.48</td>
<td>5.86206896551724</td>
</tr>
<tr>
<td>467094:647036</td>
<td>203</td>
<td>19.24</td>
<td>26.1674876847291</td>
</tr>
<tr>
<td>467143:654724</td>
<td>71</td>
<td>4.35</td>
<td>17.2676056338028</td>
</tr>
<tr>
<td>467095:647042</td>
<td>66</td>
<td>3.75</td>
<td>3.72727272727273</td>
</tr>
<tr>
<td>467111:647192</td>
<td>66</td>
<td>3.75</td>
<td>3.72727272727273</td>
</tr>
<tr>
<td>467058:646754</td>
<td>183</td>
<td>31.59</td>
<td>24.775956284153</td>
</tr>
<tr>
<td>467153:654789</td>
<td>73</td>
<td>4.97</td>
<td>13.7534246575342</td>
</tr>
</tbody>
</table>

[...]
Aggregation then Filtering

Return only those soils with a profile water storage $\geq 29$ cm

```sql
SELECT cokey, sum((bottom - top) * awc_r) as profile_awc_cm
FROM s_data
GROUP BY cokey
-- filter the results after the grouping has been done
HAVING sum((bottom - top) * awc_r) >= 29 ;
```

<table>
<thead>
<tr>
<th>cokey</th>
<th>profile_awc_cm</th>
</tr>
</thead>
<tbody>
<tr>
<td>467058:646754</td>
<td>31.59</td>
</tr>
<tr>
<td>467055:646741</td>
<td>30.1</td>
</tr>
<tr>
<td>467030:646584</td>
<td>29.42</td>
</tr>
<tr>
<td>467029:646578</td>
<td>30.28</td>
</tr>
<tr>
<td>467105:647145</td>
<td>34.5</td>
</tr>
<tr>
<td>467014:646490</td>
<td>33.78</td>
</tr>
<tr>
<td>467088:646981</td>
<td>34.5</td>
</tr>
<tr>
<td>467083:646942</td>
<td>34.5</td>
</tr>
<tr>
<td>[...]</td>
<td></td>
</tr>
</tbody>
</table>
Aggregating from a Sub-Query 1

Stack results from subsequent queries to the same table

```sql
SELECT 'Ap' as hz_type, hzname, claytotal_r
FROM chorizon where hzname like 'Ap%' and areasymbol = 'ca654'
UNION
SELECT 'Bt' as hz_type, hzname, claytotal_r
FROM chorizon where hzname like 'Bt%' and areasymbol = 'ca654'
UNION
SELECT 'C' as hz_type, hzname, claytotal_r
FROM chorizon where hzname like 'C%' and areasymbol = 'ca654';
```

<table>
<thead>
<tr>
<th>hz_type</th>
<th>hzname</th>
<th>claytotal_r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ap</td>
<td>Ap1</td>
<td>5</td>
</tr>
<tr>
<td>Ap</td>
<td>Ap2</td>
<td>10</td>
</tr>
<tr>
<td>Ap</td>
<td>Ap</td>
<td>11</td>
</tr>
<tr>
<td>Bt</td>
<td>Bt</td>
<td>47.5</td>
</tr>
<tr>
<td>Bt</td>
<td>Bt1</td>
<td>50</td>
</tr>
<tr>
<td>C</td>
<td>Cr</td>
<td>2.5</td>
</tr>
<tr>
<td>C</td>
<td>C</td>
<td>6.5</td>
</tr>
</tbody>
</table>

[...]
Compute summary stats from stacked data

```sql
SELECT hz_type, count(hz_type), avg(claytotal_r), stddev(claytotal_r)
FROM (SELECT 'Ap' as hz_type, claytotal_r
      FROM chorizon WHERE hzname like 'Ap%' and areasymbol = 'ca654'
      UNION
      SELECT 'Bt' as hz_type, claytotal_r
      FROM chorizon WHERE hzname like 'Bt%' and areasymbol = 'ca654'
      UNION
      SELECT 'C' as hz_type, claytotal_r
      FROM chorizon WHERE hzname like 'C%' and areasymbol = 'ca654'
    ) as new_data
GROUP BY hz_type;
```

<table>
<thead>
<tr>
<th>hz_type</th>
<th>count</th>
<th>avg</th>
<th>stddev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ap</td>
<td>18</td>
<td>21.388888888888889</td>
<td>16.2168717114176</td>
</tr>
<tr>
<td>Bt</td>
<td>20</td>
<td>27.05</td>
<td>11.6899642068697</td>
</tr>
<tr>
<td>C</td>
<td>24</td>
<td>18.1086956521739</td>
<td>14.0113237660076</td>
</tr>
</tbody>
</table>
Joining Tables

**Generic pattern for joining 2 tables**

```
SELECT [left-hand_table.columns], [right-hand_table.columns]
FROM left-hand_table JOIN right-hand_table
-- rule for aligning data from each table
ON [join condition]
-- optionally do more stuff after the join is complete
[WHERE clause]
[GROUP BY clause]
[ORDER BY clause]
```
Types of Joins

- **Cartesian Join**: not generally useful, returns all permutation of input rows
- **Inner Join**: most commonly used, returns rows that occur in both tables
  - 1:1 → rows missing from either table omitted
  - 1:many → rows in the left-hand table repeated
  - many:1 → rows in the right-hand table repeated (LU table)
- **Left Outer Join**: returns all records from the left-hand table, despite missing records in the right-hand table
  - 1:1, 1:many, many:1 → rows missing from right-hand table padded with NULL
- **Right Outer Join**: same as left outer join, but reversed
  - 1:1, 1:many, many:1 → rows missing from left-hand table padded with NULL
### Inner Join

**Join map unit data to component data (1:many)**

```sql
SELECT `substr`(muname, 0, 30) as muname, mapunit.mukey, cokey, compname, comppct_r
FROM mapunit JOIN component
ON mapunit.mukey = component.mukey
WHERE mapunit.mukey = '464463'
ORDER BY comppct_r DESC;
```

### Results

<table>
<thead>
<tr>
<th>muname</th>
<th>mukey</th>
<th>cokey</th>
<th>compname</th>
<th>comppct_r</th>
</tr>
</thead>
<tbody>
<tr>
<td>San Joaquin sandy loam, shall</td>
<td>464443</td>
<td>464443:641360</td>
<td>San Joaquin</td>
<td>85</td>
</tr>
<tr>
<td>San Joaquin sandy loam, shall</td>
<td>464443</td>
<td>464443:641362</td>
<td>Exeter</td>
<td>14</td>
</tr>
<tr>
<td>San Joaquin sandy loam, shall</td>
<td>464443</td>
<td>464443:641361</td>
<td>Unnamed, ponded</td>
<td>1</td>
</tr>
</tbody>
</table>
## Left-Outer Join

**Generate a listing of restrictive features for a single map unit**

```sql
SELECT mukey, component.cokey, compname, comppct_r, reskind, reshard
FROM component
LEFT JOIN corestrictions
ON component.cokey = corestrictions.cokey
WHERE mukey = '464443'
ORDER BY comppct_r DESC;
```

### Results of a left-outer join

<table>
<thead>
<tr>
<th>mukey</th>
<th>cokey</th>
<th>compname</th>
<th>comppct_r</th>
<th>reskind</th>
<th>reshard</th>
</tr>
</thead>
<tbody>
<tr>
<td>464443</td>
<td>464443:641360</td>
<td>San Joaquin</td>
<td>85</td>
<td>Duripan</td>
<td>Indurated</td>
</tr>
<tr>
<td>464443</td>
<td>464443:641362</td>
<td>Exeter</td>
<td>14</td>
<td></td>
<td></td>
</tr>
<tr>
<td>464443</td>
<td>464443:641361</td>
<td>Unnamed, ponded</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
SELECT mukey, mu_area_frac, taxgrtgroup, hd.cokey as id, top, bottom, prop
FROM
(
    SELECT cd.mukey, cokey, taxgrtgroup, (comppct_r::numeric / 100.0) * mu_area as mu_area_frac
    FROM
        (SELECT mukey, cokey, taxgrtgroup, comppct_r, comppct_r::numeric / 100.0 * mu_area as mu_area_frac
            FROM
                (SELECT mukey, cokey, comppct_r, taxgrtgroup
                    FROM component
                    WHERE areasymbol = 'ca654'
                    AND taxgrtgroup is not NULL)
            AS cd
        ) AS cd
    JOIN
        (SELECT mukey, sum(ST.Area(wkb.geometry)) as mu_area
            FROM mapunit_poly
            WHERE areasymbol = 'ca654'
            GROUP BY mukey
        ) AS mu_areas
    ON cd.mukey = mu_areas.mukey
) AS comp_wts
JOIN
(
    SELECT cokey, hzdept_r as top, hzdepb_r as bottom, claytotal_r as prop
    FROM chorizon
    WHERE om_r is not null
    AND areasymbol = 'ca654'
) AS hd
ON comp_wts.cokey = hd.cokey
ORDER BY mukey, id, top ASC;