Introduction to SQL

with examples from digital soil survey data

D. E. Beaudette

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

debeaudette@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.

[1] http://en.wikipedia.org/wiki/SQL

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 $\rightarrow$ 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 langauges. Extensions provide the ability to "call" external functions from other common programming languages, entirely within the context of the database.

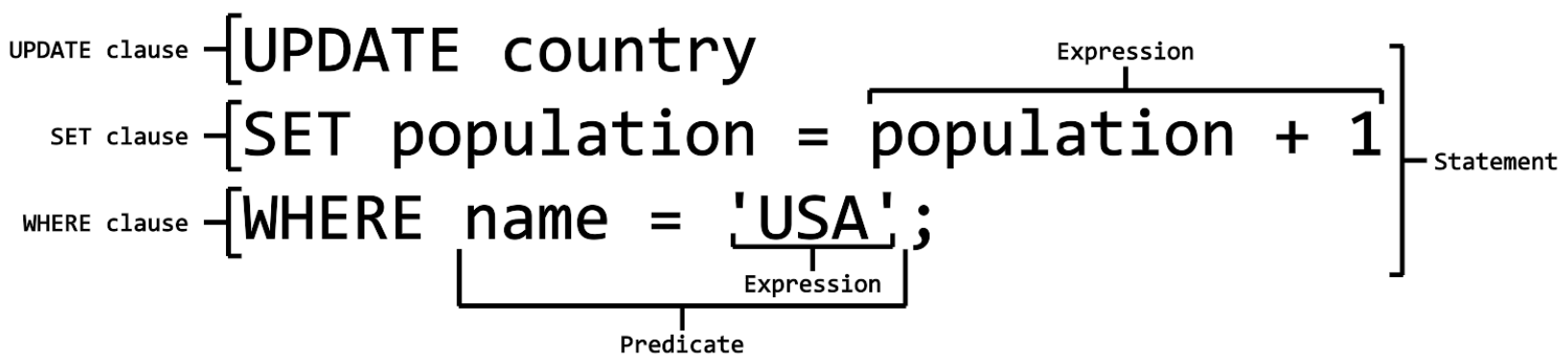
Some Examples

- "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



[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

cokey	hzname	hzdept_r	hzdepb_r
467038:646635	Ap	0	18
467038:646635	Bw	18	41
467038:646635	Bk1	41	69
467038:646635	Bk2	69	109
467038:646635	Bk3	109	145
467038:646635	Bk4	145	183

INSERT/UPDATE/DELETE Statements

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[3]

Altering Table Structure

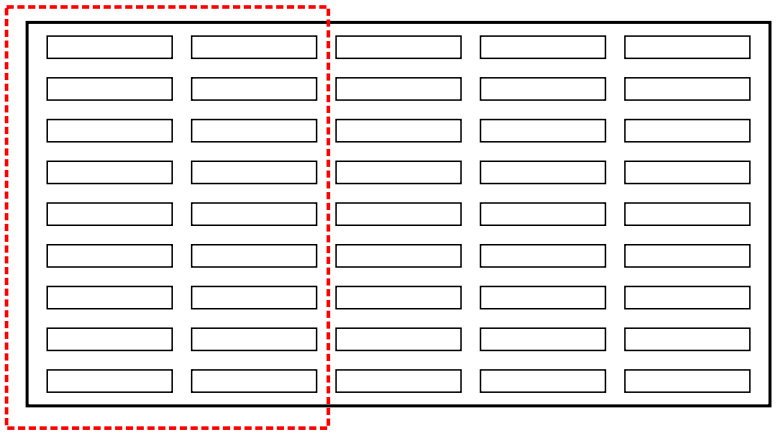
```
-- add a new column
ALTER TABLE chorizon ADD COLUMN hydrophobicity_index integer;
-- now remove the column
ALTER TABLE chorizon DROP COLUMN hydrophobicity_index integer;
```

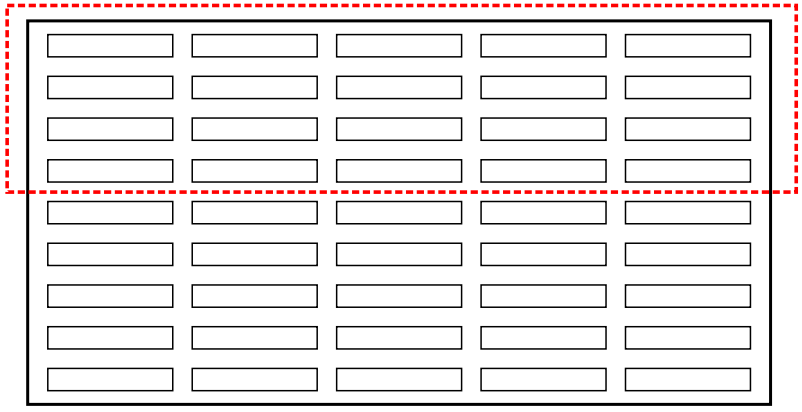
Altering Column Definitions

```
-- rename a column
ALTER TABLE chorizon RENAME COLUMN claytotal_r TO clay;
-- change the column's datatype (be careful!)
ALTER TABLE chorizon ALTER COLUMN clay TYPE numeric;
-- do not allow NULL values in a column
ALTER TABLE chorizon ALTER COLUMN clay SET NOT NULL;
-- do not allow values over 100%
ALTER TABLE chorizon ALTER COLUMN clay CHECK (clay <= 100) ;
```

[3] http://www.postgresql.org/docs/8.3/static/sql-altertable.html

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

```
SELECT * from chorizon;
```

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

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

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

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

 hz_interval
--------------
 0-5
[...]
```

Filtering by Row

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

```
SELECT hzname, sandtotal_r, silttotal_r, claytotal_r
FROM chorizon
WHERE claytotal_r >= 40 ;
```

hzname	sandtotal_r	silttotal_r	claytotal_r
Bt2	26.1	28.9	45
Bt2	26.1	28.9	45
Bw	5.3	44.7	50
Az	7.2	47.8	45
Cz	26.1	28.9	45
Bw1	30	22.5	47.5
[...]			

...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 DISTICT hzname
FROM chorizon
WHERE hzname LIKE '%A%'
AND areasymbol = 'ca653';

 hzname
________
 A
 A1
 A2
 A3
 AB
 ABt
 Ad
 Agb
 Anz
 Ap
 Ap1
 [...]
```

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' ;
```

mukey	comppct_r	majcompflag	compname
459154	100	Yes	Water
459210	85	Yes	Balcom
459212	45	Yes	Balcom
459212	40	Yes	Dibble
459213	85	Yes	Brentwood

This is also possible

```
SELECT mukey, comppct_r, majcompflag, compname
from component
where mukey in (SELECT mukey from [...])
AND majcompflag = 'Yes' ;
```

ORDER BY

Return some data ordered by multiple strata

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

mukey	cokey	comppct_r	top	bottom	db	om
459210	459210:623942	85	0	61	1.6	0.75
459210	459210:623942	85	61	94	1.6	0.25
459210	459210:623942	85	94	104		
459212	459212:623950	45	0	51	1.6	0.75
459212	459212:623950	45	51	76	1.6	2.5
459212	459212:623950	45	76	86		
459212	459212:623951	40	0	10	1.66	0.75
459212	459212:623951	40	10	76	1.68	0.25
459212	459212:623951	40	76	86		
459213	459213:623953	85	0	25	1.81	0.75
459213	459213:623953	85	25	89	1.74	0.25
459213	459213:623953	85	89	152	1.78	0.25

Making a New Table from a SELECT Statement

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

```
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

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

[...]

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

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

[...]

Aggregation then Filtering

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

```
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 ;
```

cokey	profile_awc_cm
467058:646754	31.59
467055:646741	30.1
467030:646584	29.42
467029:646578	30.28
467105:647145	34.5
467014:646490	33.78
467088:646981	34.5
467083:646942	34.5
[...]	

Aggregating from a Sub-Query 1

Stack results from subsequent queries to the same table

```
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' ;
```

hz_type	hzname	claytotal_r
Ap	Ap1	5
Ap	Ap2	10
Ap	Ap	11
Bt	Bt	47.5
Bt	Bt1	50
C	Cr	2.5
C	C	6.5

[...]

Aggregating from a Sub-Query 2

Compute summary stats from stacked data

```
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;
```

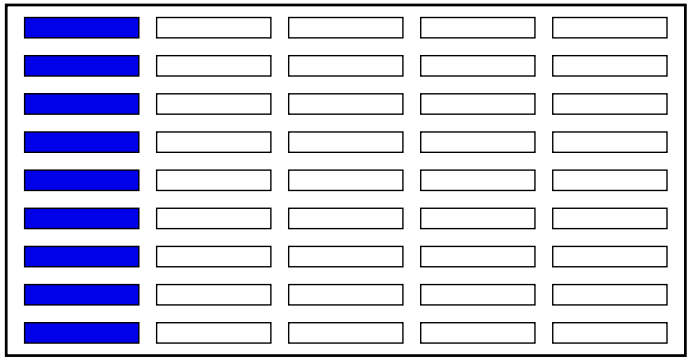
hz_type	count	avg	stddev
Ap	18	21.3888888888889	16.2168717114176
Bt	20	27.05	11.6899642068697
C	24	18.1086956521739	14.0113237660076

Joining Tables

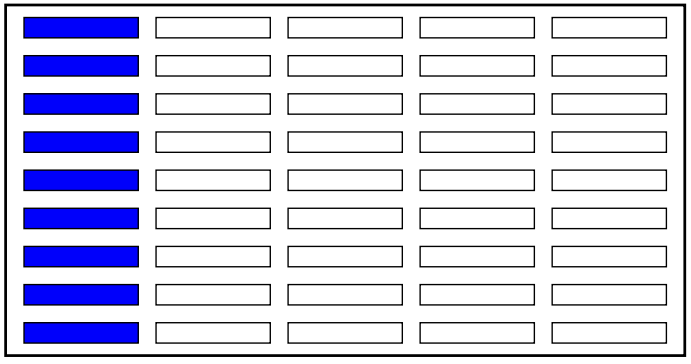
Generic pattern for joining 2 tables

```
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]
-- optionaly do more stuff after the join is complete
[WHERE clause]
[GROUP BY clause]
[ORDER BY clause]
```

left-hand table



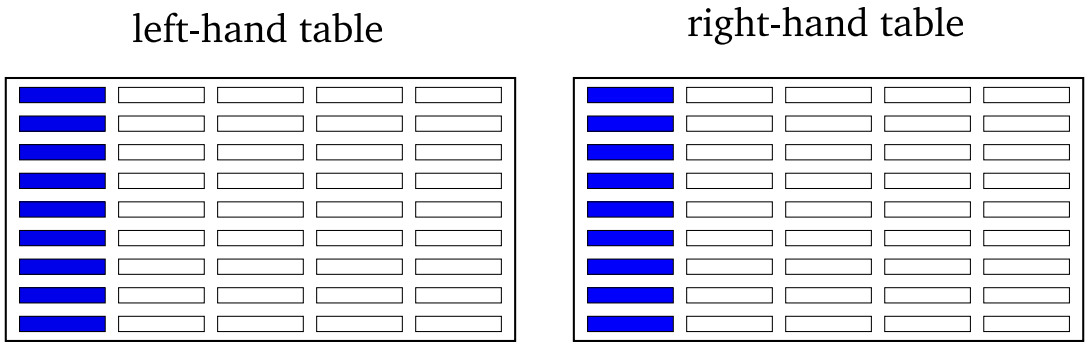
right-hand table



Joins

Types of Joins

- Cartiesian 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 ommitted
  - 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



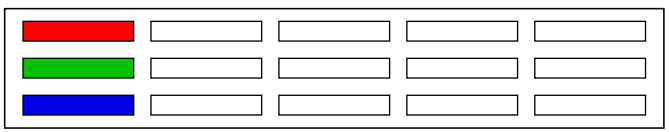
Inner Join

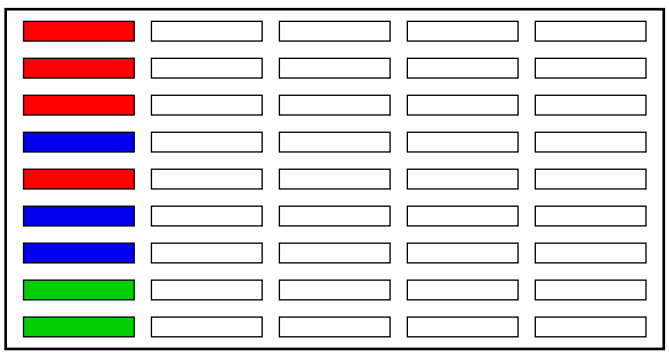
Join map unit data to component data (1:many)

```
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

muname	mukey	cokey	compname	comppct_r
San Joaquin sandy loam, shall	464443	464443:641360	San Joaquin	85
San Joaquin sandy loam, shall	464443	464443:641362	Exeter	14
San Joaquin sandy loam, shall	464443	464443:641361	Unnamed, ponded	1





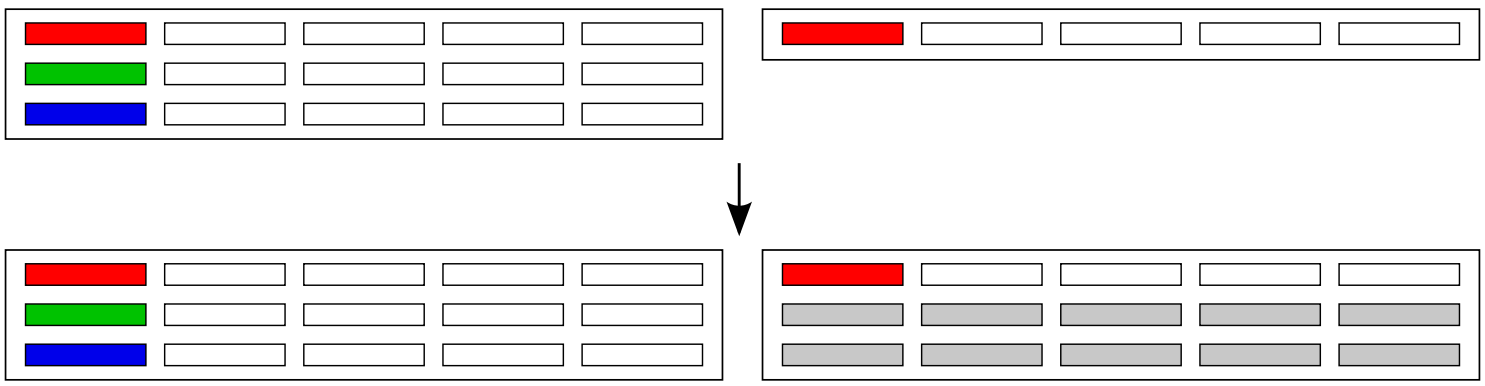
Left-Outer Join

Generate a listing of restrictive features for a single map unit

```
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

mukey	cokey	compname	comppct_r	reskind	reshard
464443	464443:641360	San Joaquin	85	Duripan	Indurated
464443	464443:641362	Exeter	14		
464443	464443:641361	Unnamed, ponded	1		



Joins to Nested Sub-Queries

```
SELECT mukey, mu_area_frac, taxgrtgroup, hd.cokey as id, top, bottom, prop
FROM
(
        -- component weights, in the form of area fractions
        SELECT cd.mukey, cokey, taxgrtgroup, (comppct_r::numeric / 100.0) * mu_area as mu_area_frac
        FROM
        (
                -- component keys and percentages
                SELECT mukey, cokey, comppct_r, taxgrtgroup
                from component
                where areasymbol = 'ca654'
                and taxgrtgroup is not NULL
                ) as cd
        JOIN
        (
                -- map unit areas by mukey
                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
-- regular join will throw out all components without horizon data
JOIN
(
        -- horizon level data
        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;
```