# Grouping Zoning Applicants via Lookup Tables

This sequence of SQL queries uses Clark Broida's Zoning database[*] to illustrate strategies for combining local information with other people's data.

The 'ZONING' database contains information about all zoning variances formally requested in Boston during two+ years in the mid-1980s -- the heyday of the Boston's 80s boom. Suppose we wish to compare the types of zoning variances filed by public and private institutions and by individuals during that time. The 'applicant' field stores the name of zoning variance applicants and is often enough to tell us whether it was a company, the city, a non-profit organization, or an individual. For example, here's everyone with 'Boston' in their name:

```
COLUMN fname FORMAT A15

  select applicant, fname, count(*) variances
    from zoning
   where applicant like '%BOSTON%'
group by applicant, fname
order by applicant, fname;

APPLICANT                      FNAME            VARIANCES
------------------------------ --------------- ----------
BANK OF BOSTON                                          1
BOSTON CENTER FOR THE ARTS                              1
BOSTON CHATAQUA PROPERTIES,INC                          1
BOSTON COLLEGE HIGH SCHOOL                              1
BOSTON COLLEGE, TRUSTEES OF                             1
BOSTON DESIGN CENTER                                    1
BOSTON FOOD COOPERATIVE INC.                            1
BOSTON HOUSING AUTHORITY                                3
BOSTON LOCK & SAFE COMPANY                              1
BOSTON PRESERVATION ALLIANCE                            1
BOSTON SCHOOL HOUSE ASSOCIATES                          1
...
```

Here's everyone with 'university' in their name:

```
 SELECT applicant, fname, count(*) variances
   FROM zoning
   WHERE applicant LIKE '%UNIVERSITY%'
GROUP BY applicant, fname;

APPLICANT                      FNAME            VARIANCES
------------------------------ --------------- ----------
BOSTON UNIVERSITY                                       1
BOSTON UNIVERSITY, TRUSTEES                             1
BOSTON UNIVERSITY, TRUSTEES OF                          1
```

[*] Kindly refer to the Tools section

```
HARVARD UNIVERSITY                                     1
NORTHEASTERN UNIVERSITY                                3
SHOWA WOMEN'S UNIVERSITY OF JA                         1
SUFFOLF UNIVERSITY                                     1
```
Note that Boston University submitted 3 variances and their 'name' is spelled differently each time. We could standardize the spelling of Boston University in the zoning table. But, if the zoning table was maintained by another agency and updated periodically (to report hearing results, subsequent design review outcomes, etc.) we do not want to make any changes to our copy of the 'master' zoning table. Even if we are fixing spelling errors, we would have to make the corrections all over again each time a new 'official' version of the zoning table arrived.

Let's take a look at some of the data and try to use our understanding of relational database management to find a way in which we can handle data errors and re-groupings.

```
  select applicant, count(*) variances
    from zoning
group by applicant
  having count(*) > 3
order by count(*) DESC;

applicant                        variances

SMITH                                   11
BRA                                     10
SULLIVAN                                 6
KELLY                                    6
DOHERTY                                  6
DUMBAUGH                                 5
LEE                                      5
MCCARTHY                                 5
WONG                                     4
O'CONNELL                                4

CLIFFORD                                 4
BARRY                                    4
DRISCOLL                                 4
CARROLL                                  4
WILLIAMS                                 4
```
Out of the 1801 zoning variances reported in the table, only 192 applicant names are repeated! And, the top-15 applicant list is dominated by individual names. Only the 'BRA', the Boston Redevelopment Authority, looks like it isn't the name of an individual.

But, the earlier queries already showed us that Boston University had filed 3 variances with a different spelling of their name each time. In fact, a little digging shows us that the 'City of Boston' is listed only once, but the Public Facilities Department (of the City) filed 5 or 6 variances under 3 or 4 different spellings, depending upon whether the Public Facilities Commission is something different from PFD! Shortly, we'll assume that it is not.

```
 select applicant, count(*) variances
   from zoning
```

```
  where applicant like '%FACILIT%'
group by applicant;

applicant                       (count(*))

PUBLIC FACILITIES DEPT                  2
PUBLIC FACILITIES DEPARTMENT            2
PUBLIC FACILITES COMMISSION             1
PUBLIC FACILITIES CITY OF BOST          1
```

With a more little time and effort, we could do more such queries and gradually develop a much sharper sense of which applicants are institutions and who are the big players. BY USING SOME OF WHAT WE HAVE LEARNED ABOUT RELATIONAL DATA MODELS, WE CAN KEEP THESE CHANGES IN A SEPARATE TABLE AND ACCUMULATE THE RESULTS OF OUR EFFORT OVER TIME WITHOUT DISTURBING THE 'OFFICIAL' TABLE.

Let's start by creating a lookup table to store the (first and last) name of applicants, the corrected applicant name and a yet-to-be-determined grouping where we will store 'public', 'private', 'individual', etc.. We could create the table explicitly and then 'insert' rows, or create the table as the 'select' statement. We show both ways below. When creating the table explicitly, be sure that the datatypes are an exact match or you may have trouble joining the tables later. When creating as a 'select', you must rename the duplicate copy of 'applicant' and set the new field to a long string of the desired width.

```
CREATE TABLE apptype
( fname varchar(20),
  applicant varchar(30),
  newapp varchar(30),
  agroup varchar(12));

INSERT INTO apptype
SELECT DISTINCT fname, applicant, applicant, 'UNKNOWN'
  FROM zoning;

CREATE TABLE apptype AS
SELECT DISTINCT fname, applicant, applicant newapp,
       'xxxxxxxxxxxx' agroup
  FROM zoning;
UPDATE apptype SET agroup = 'UNKNOWN';

create index app on apptype(applicant);
```

1702 rows are inserted/indexed.

In class and lab, we developed several SQL queries using the zoning table and the new apptype table to determine the number of variances and total existing square footage for property owners that are the city, local universities, Trusts, etc. Note that the zoning table contains a field, 'fname' (for first name), that is blank if the applicant is not an individual.

Now, let's build some 'rules' that will store the knowledge we wish to accumulate about spelling corrections in applicant names and group membership.

```
update apptype
  set newapp = 'BOSTON, PFD' where applicant like '%FACILIT%';
update apptype
  set newapp = 'BOSTON UNIVERSITY'
  where applicant like '%BOSTON UNIVERSITY%';
```

These two 'updates' standardize the name used to represent PFD and Boston University by replacing what is stored in the 'newapp' field. Let's run some queries to see the results 'before' and 'after' regrouping via 'newapp':

*BEFORE:*

```
select z.applicant, count(*) variances
  from zoning z
 where z.applicant like '%UNIVERSITY%'
 GROUP BY z.applicant;

APPLICANT                       VARIANCES
------------------------------ ----------
BOSTON UNIVERSITY                       1
BOSTON UNIVERSITY, TRUSTEES             1
BOSTON UNIVERSITY, TRUSTEES OF          1
HARVARD UNIVERSITY                      1
NORTHEASTERN UNIVERSITY                 3
SHOWA WOMEN'S UNIVERSITY OF JA          1
SUFFOLF UNIVERSITY                      1
```

*AFTER regrouping:*

```
select a.newapp, count(*) variances
  from zoning z, apptype a
 where z.applicant = a.applicant and
       z.fname = a.fname and
       a.applicant like '%UNIVERSITY%'
 GROUP BY a.newapp;

NEWAPP                          VARIANCES
------------------------------ ----------
BOSTON UNIVERSITY                       3
HARVARD UNIVERSITY                      1
NORTHEASTERN UNIVERSITY                 3
SHOWA WOMEN'S UNIVERSITY OF JA          1
SUFFOLF UNIVERSITY                      1
```

```
Now, let's run some queries that use the apptype table to check which
applicants have the most square footage under zoning review.
 select fname, applicant, sum(existsqft) sqft, count(*) variances
   from zoning
group by fname, applicant
  having count(*) >= 3
order by 4 desc;

FNAME                APPLICANT                            SQFT
VARIANCES
-------------------- ------------------------------ ---------- --------
--
                     BRA                               1362955
9
```

```
CHARLES C.          DUMBAUGH                                11059
5
                    BOSTON HOUSING AUTHORITY                   -2
3
KEVIN               CARROLL                                     0
3
                    GOLD ASSOCIATES                          7997
3
                    NORTHEASTERN UNIVERSITY                111929
3
```
This list uses none of our re-grouped data and shows few owners of multiple variance requests. (We'll ignore the problems with negative (unknown) square footages for now.) Next, we'll join to the 'apptype' and run the same query but group by the 'newapp' field.
```
 select a.fname, a.newapp, sum(existsqft) sqft, count(*) variances
   from zoning z, apptype a
  where z.applicant = a.applicant and
        z.fname = a.fname
group by a.fname, a.newapp
  having count(*) >= 3
order by 4 desc;

FNAME               NEWAPP                                   SQFT
VARIANCES
-------------------- ------------------------------ ---------- --------
--
                    BRA                                   1362955
9
                    BOSTON, PFD                            159722
5
CHARLES C.          DUMBAUGH                                11059
5
                    BOSTON HOUSING AUTHORITY                   -2
3
KEVIN               CARROLL                                     0
3
                    NORTHEASTERN UNIVERSITY                111929
3
                    GOLD ASSOCIATES                          7997
3
                    BOSTON UNIVERSITY                      184163
3
```
The standardization of PFD and Boston University names moves them onto our multiple-variance list. But the 'sqft' column reminds us that square footage is not always known -- remember that a code of '-1' is used for 'missing'. With modern RDBMS packages, such cases would be encoded as NULL. We could rerun the query excluding those zoning variance cases where square footage is not known.
```
  select a.fname, a.newapp, sum(existsqft) sqft, count(*) variances
   from zoning z, apptype a
  where z.applicant = a.applicant and
        z.fname = a.fname and
        existsqft > 0
group by a.fname, a.newapp
having count(*) >= 3
order by 4 desc;
```

```
FNAME      NEWAPP                               SQFT  VARIANCES
---------- ------------------------------ ---------- ----------
           BOSTON, PFD                        159722          4
CHARLES C. DUMBAUGH                            11060          4
           BRA                               1362958          3
```

Wow! Most of the multi-variance owners drop out when we consider only the cases where square footage is reported. That probably makes sense since it's know for most residential property with a small lot and single individual owner. But who knows the size of a public housing development when they fill out a zoning variance application for the BHA.

---

Instead of pursuing this line of reasoning further where we are only correcting the misspelling of owner names, let's use the same idea to CATEGORIZE TYPE OF OWNERSHIP by updating the values stored in the 'agroup' field of our 'apptype' lookup table.

```
update apptype
  set agroup = 'INSTITUTION' where fname = ' ';

update apptype
  set agroup = 'TRUST'
 where applicant like '%TRUST%';

update apptype
  set agroup = 'UNIVERSITY'
 where applicant like '%UNIVERSITY%';

update apptype
  set agroup = 'CITY'
 where (applicant like '%CITY%' and applicant like '%BOSTON%')
   or applicant like '%FACILIT%';

commit;
/* commit these changes so they won't be rolled back
  if you run an SQL with a syntax error  */
```

The first of these four 'updates' sets the 'agroup' to 'INSTITUTION' wherever the firstname was blank in the original data. The next three 'updates' reset 'agroup' for records that appear to be universities, trusts, or the city of Boston.

```
select agroup, sum(existsqft) sqft, count(*) variances
from zoning z, apptype a
where z.applicant = a.applicant and
      z.fname = a.fname
group by agroup
having count(*) >= 3
order by 3 desc;

AGROUP               SQFT  VARIANCES
```

```
------------ ---------- ----------
UNKNOWN         4555769       1268
INSTITUTION    12636378        346
TRUST           1994811        170
UNIVERSITY       644835          9
CITY             159721          8
```

The square footage associated with the 'institutional' owned parcels requesting variances is larger than that of the 'unknown' group that comprises mainly individually owned parcels. In fact, the 'trust' and 'university' cases that were separated from the other institutional cases amount to more than half the square footage of the individually owned cases.

Now, let's add conditions and qualifiers to the same basic SQL to examine further some of the differences that might exist in zoning variance characteristics across the five broad categories of ownership that we have identified. E.G., here's a table that looks only at residentially zoned property and computes the fraction of cases that involved floor area ratio violations (among all cases where FAR151 was not missing).

```
COLUMN far_percent FORMAT 999.99

  select a.agroup, 100*avg(far151-1) FAR_percent,
         sum(existsqft) sqft, count(*) variances
    from zoning z, apptype a
   where z.applicant = a.applicant and
         z.fname = a.fname and
         substr(EXISTZONIN,1,1) in ('R', 'S', 'H') and
         far151 > 0
group by a.agroup
order by 4 desc;

AGROUP       FAR_PERCENT       SQFT  VARIANCES
------------ ----------- ---------- ----------
UNKNOWN            34.36    2880837        972
INSTITUTION        33.33    6738403        174
TRUST              45.00     808701        100
CITY                 .00      88882          5
UNIVERSITY           .00     348743          3
```

Hmm, a high percentage of residential variances requested floor area ratio violations and the 'trust' properties were more likely than average to want this.

Now let's redo this query but instead of looking at floor area ratios violations, let's look at front/side/rear yard restrictions. (See the appendix below for various descriptive statistics regarding these codes.)

```
create table t1yard as
select distinct casenumber, RYARD201, FYARD181, SYARD191,
       0 combo
  from zoning;

update t1yard
   set combo = 1
 where (RYARD201 = 2 or FYARD181 = 2 or SYARD191 = 2);
```

```
  select combo, count(*) variances
    from t1yard
group by combo
order by combo;

     COMBO  VARIANCES
---------- ----------
         0       1085
         1        713   <== 40% involve yard violations

COLUMN yard_percent FORMAT 999.99

  select a.agroup, 100*avg(combo) yard_percent,
         sum(existsqft) sqft, count(*) variances
    from zoning z, apptype a, t1yard t
   where z.applicant = a.applicant and
         z.fname = a.fname and z.casenumber = t.casenumber and
         substr(EXISTZONIN,1,1) in ('R', 'S', 'H')
group by a.agroup
order by 4 desc;

AGROUP       YARD_PERCENT       SQFT  VARIANCES
------------ ------------ ---------- ----------
UNKNOWN             43.92    2914756        979
INSTITUTION         33.90    8579687        177
TRUST               32.35     816726        102
CITY                60.00      88882          5
UNIVERSITY            .00     348743          3
```
Hmm, institutionally owned (and trusts) are less likely to have front/side/rear yard restrictions! Given the number of cases, that difference is significant.

At this point, we might remember from class that there are several duplicate casenumber in the zoning table and we may want to exclude those cases (since they involve some form of mistakes) to avoid biasing our results. To do this, we illustrate the use of side tables to handle outliers and special cases by creating a table with the casenumbers for all cases that have duplicate rows in the zoning table. Since the characteristics differ and we don't know which set is 'correct', we'll probably want to exclude these case numbers from many of our SQL queries.

```
create table t1double as
  select casenumber, count(*) cases
    from zoning
group by casenumber
  having count(*) > 1;
```
Now, redo the previous breakdown of variances by ownership category, excluding the 8 duplicated cases:
```
  select a.agroup, 100*avg(combo) yard_percent,
         sum(existsqft) sqft, count(*) variances
    from zoning z, apptype a, t1yard t
   where z.applicant = a.applicant and
         z.fname = a.fname and z.casenumber = t.casenumber and
         substr(EXISTZONIN,1,1) in ('R', 'S', 'H') and
         z.casenumber NOT IN (select casenumber from t1double)
```

```
group by a.agroup
order by 4 desc;

AGROUP       YARD_PERCENT       SQFT  VARIANCES
------------ ------------ ---------- ----------
UNKNOWN             43.92    2819027        970
INSTITUTION         33.53    4897117        173
TRUST               31.63     800676         98
CITY                60.00      88882          5
UNIVERSITY            .00     348743          3
```

Not too much change in the results, but it does become a useful technique to exclude cases that you've identified as outliers.

Finally, let's try to find another condition that varies between 'trust', 'institution' and 'unknown' (mostly individuals). What about lotsize restrictions? We can group the four relevant variables in the same fashion that we just did for front/side/rear yard violations.

```
create table t1lot as
select distinct casenumber,
       LOTSZE141, LOTSZE142, LOTSZE143, LOTSZE144, 0 combo
  from zoning;

update t1lot
   set combo = 1
 where ( LOTSZE141 = 2 or LOTSZE142 = 2 or LOTSZE143 = 2
       or LOTSZE144 = 2);

  select combo, count(*) variances
    from t1lot
group by combo
order by combo;

     COMBO  VARIANCES
---------- ----------
         0       1103
         1        694   <== 39% of variances involved lotsize

COLUMN lot_percent FORMAT 999.99

  select a.agroup, 100*avg(combo) lot_percent,
         sum(existsqft) sqft, count(*) variances
    from zoning z, apptype a, t1lot t
   where z.applicant = a.applicant and
         z.fname = a.fname and z.casenumber = t.casenumber and
         substr(EXISTZONIN,1,1) in ('R', 'S', 'H') and
         z.casenumber NOT IN (select casenumber from t1double)
group by a.agroup
order by 4 desc;

AGROUP       LOT_PERCENT       SQFT  VARIANCES
------------ ----------- ---------- ----------
UNKNOWN            49.79    2819027        970
INSTITUTION        33.53    4897117        173
TRUST              41.84     800676         98
CITY               40.00      88882          5
```

```
UNIVERSITY            .00     348743          3
```
Hmm, this time the difference between unknown (i.e., individually owned properties) and institutionally owned properties is even larger. Institutionally owned residential properties are much less likely to seek variances for lotsize violations.

---

Of course, this breakdown of ownership categories is only a beginning since many other 'rules' could be constructed to sort out and better categorize the class of ownership. As we learn more and more about the applicants, we can add more 'update' rules.

In this way, we can accumulate our knowledge gradually without disturbing the 'official' files and without altering the basic queries. As we add more 'knowledge' to our lookup table, the same query will produce different (better and better) results. We never have to make any changes to the original zoning table. When an updated version of the zoning table becomes available, we can replace our old one with the new table, and still use the same old queries and all our knowledge in the lookup table. We'll do a bit of this in the homework.

**FOOTNOTE:** More complications can arise if the original zoning table undergoes major changes in its structure or content from one version to the next. But, we can address many of these complications with other slightly more complex strategies. For example, when a new, updated zoning table arrives, we can run queries to identify all the names in the new zoning table that don't show up in our old 'apptype' table (and vice versa).

---

## APPENDIX: Examining Front/Side/Rear Yard Violations

### OFFPKG231

```
  select SYARD191, count(*) variances
    from zoning
group by SYARD191
order by SYARD191;

  SYARD191  VARIANCES
---------- ----------
        -1          8
         1       1505
         2        288

  select FYARD181, count(*) variances
    from zoning
group by FYARD181
order by FYARD181;

  FYARD181  VARIANCES
---------- ----------
```

```
        -1          8
         1       1471
         2        322

  select RYARD201, count(*) variances
    from zoning
group by RYARD201
order by RYARD201;

  RYARD201  VARIANCES
---------- ----------

        -1          8
         1       1327
         2        466
```
Only 8 cases had missing values for each of these front/side/rear variance requests, and the following query shows they were the same 8 cases:
```
select count(*) from zoning
 where  RYARD201 <=0 or FYARD181 <=0 or SYARD191 <=0;

  COUNT(*)
----------
         8
```
Of the cases where the front/side/rear variance requests were recorded, here's the breakdown by various combinations:
```
select count(*) from zoning
 where RYARD201 = 2 or FYARD181 = 2 or SYARD191 = 2;

  COUNT(*)
----------
       713

select count(*) from zoning
 where  RYARD201 >0 and FYARD181 >0 and SYARD191 >0;

  COUNT(*)
----------
      1793

  select (RYARD201-1 + FYARD181-1 + SYARD191-1) total,
         count(*) variances
    from zoning
   where RYARD201 >0 and FYARD181 >0 and SYARD191 >0
group by (RYARD201-1 + FYARD181-1 + SYARD191-1);

     TOTAL  VARIANCES
---------- ----------
         0       1080
         1        422
         2        219
         3         72
```