**Getting Permit Data Ready for Owner Group Definitions**

SQL script for 2010 and beyond:

SELECT business\_ids.ap\_year AS ap\_year,

bo.person\_id AS person\_id,

cast(business\_ids.vp\_num AS VARCHAR(255)) AS permit

FROM (

SELECT ap\_nums.ap\_year,

ap\_nums.vp\_num,

ap\_nums.min\_ap\_num,

(SELECT min(start\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS start\_date,

(SELECT max(end\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS end\_date,

vo.business\_id

FROM (

SELECT ap\_year, vp\_num, min(ap\_num) as min\_ap\_num

FROM PERMIT.vps\_fishery\_ner /\*\*\*\* NOTE: A WHERE CLAUSE COULD BE INSERTED AFTER THIS LINE LIMITING PERMIT TYPES \*\*\*\*/

GROUP BY ap\_year, vp\_num

) ap\_nums

LEFT JOIN PERMIT.VPS\_OWNER vo ON vo.ap\_num = ap\_nums.min\_ap\_num AND vo.rank = 1

) business\_ids

LEFT JOIN PERMIT.BUSINESS\_OWNER bo ON business\_ids.business\_id = bo.business\_id AND bo.start\_date <= business\_ids.end\_date

AND (bo.end\_date IS NULL OR bo.end\_date >= business\_ids.start\_date)

WHERE business\_ids.ap\_year >= 2010

What this script does:

Step 1: Pull all the vessels (vp\_num) from vps\_fishery\_ner that had any kind of fishing permit in 2010 (ap\_year) and beyond. Select the minimum ap\_num for each vp\_num. Choosing the minimum ap\_num gets the ownership status at the point in the year when the first permit application was made. Changes may have occurred in subsequent applications (higher ap\_nums) but we need to choose one point in time for each vessel. Also pull the variables start\_date and end\_date that will be used in step 5. These variables show the beginning and end of the time period for which each ap\_num is associated.

Step 2: Get the business\_id associated with every ap\_num within the appropriate ap\_year. Business\_id is found in the vps\_owner table. Only choose business\_ids where the variable “rank” is equal to 1. This is per Ted Hawes -- rank 1 is the primary owner, rank 2 is the secondary owner, etc.

Step 3: Merge step 1 with step 2 on the ap\_nums from step 1 only. This is because vp\_num is largly missing in the vps\_owner table. However, ap\_num is a unique transaction within a given year. So, the vp\_num from step 1 will carry over in this merge. Drop all other non-matched ap\_nums and business\_ids (the ones from step 2 that don’t match those in step 1).

Step 4: Get all the person\_ids, business\_ids, and start/end dates for all entries in the business\_owner table. The variables start\_date and end\_date in the business\_owner table are different from the ones in vps\_fishery\_ner. The date variables in the business\_owner table define the time period a person\_id is associated with a business\_id. IMPORTANT NOTE: if end\_date = null then that person\_id is still associated with the business\_id.

Step 5: Merge step 3 with step 4. The start/end dates associated with the person\_ids (from business\_owner, step 4) need to be within the start/end dates associated with the minimum ap\_nums carried forward in steps 1 thru 3. For example, business\_id #500 (Cod Fishing Inc) could have been in existence starting in 2010 to present but person\_id (Rob) was a shareholder of Cod Fishing Inc for only the first half of 2010 (start\_date = Jan 2010, end\_date = Jul 2010). The two other person\_ids (Steve and Jack) were shareholders since Jan 2010 to present day (end\_date = null). So if the minimum ap\_num for this particular vessel (vp\_num) has start/end dates that span any part of the first half of 2010 (doesn’t have to line up exactly), Rob’s person\_id should then be listed along with Steve’s and Jack’s. Otherwise, list Steve and Jack only.

This SQL script queries permits from all fisheries and so subsequent owner group definitions would then be based on the resulting data set. The script can easily be modified to limit it to a particular set of permit plans and/or categories (see note in script above).

The end result is three variables: ap\_year, permit, person\_id. If there are more than one person\_id values associated with a permit number at the time the first permit application was made within a fishing year (dictated by the choice of minimum ap\_num), then there will be multiple entries of the variable “permit”, each with different person\_id variables.

SQL script for pre-2010:

SELECT min\_ap\_nums.ap\_year, ohp2.person\_id, min\_ap\_nums.permit

FROM (SELECT ap\_year, permit, min(ap\_num) as min\_ap\_num

FROM PERMIT.owner\_history\_pids

WHERE ap\_year <= 2009

and person\_id is not null and permit is not null

AND ap\_num IN (SELECT ap\_num FROM PERMIT.vps\_fishery\_ner WHERE plan = 'MUL' AND cat in ('A','B','C','D','E','F','G','HA'))

GROUP BY ap\_year, permit) min\_ap\_nums

LEFT JOIN PERMIT.OWNER\_HISTORY\_PIDS ohp2 ON min\_ap\_nums.AP\_YEAR = ohp2.ap\_year AND min\_ap\_nums.permit = ohp2.permit AND min\_ap\_nums.min\_ap\_num = ohp2.ap\_num

What this script does:

Prior to 2010, owner data is only available for groundfish and scallop permit holders. The SQL script above limits the selections to limited access groundfish permit holders, as an example.

The SQl script essentially does the same thing as the post-2009 script but it pulls the person\_id variables from the owner\_history\_pids table. This table only contains pre-2010 data (back to 1994) and the vps\_owner and business\_owner tables referenced in the post-2009 script contain only post-2009 data.

In owner\_history\_pids, person\_id is accessed directly. There is no business\_id. The other distinction of owner\_history\_pids is that there are cases where a person\_id is given for a subset of ap\_nums only. This may or may or may not correspond with the minimum ap\_num pulled from the vps\_fishery\_ner table. In these cases, the minimum ap\_num from the owner\_history\_pids table is used and linked to an existing ap\_num from the vps\_fishery\_ner table. Otherwise, as in the post-2009 script, minimum ap\_num is first pulled from vps\_fishery\_ner then linked to that ap\_num in the owner\_history\_pids table.

**Defining Owner Groups**

Group Definition #1 Level (RFA Level):

Group Definition #1 is also referred to as the RFA definition because this is the definition the Social Sciences Branch and the Regional Office are currently discussing as a definition for use in determining large vs. small businesses in Regulatory Flexibility Act analyses.

Every person associated with a business ID is assigned a person ID. Through the use of person IDs associated with each MRI (via the fishing permit), unique combinations of owners are categorized into a single owner group. This is best illustrated with an example:

If the same two owners own separate MRIs, they count as one entity. If one owner is involved in three different MRIs with three different owners, that counts as three entities. If a MRI is owned by a single owner, that counts as one entity. To illustrate this concept, Table 1 shows the interrelationship between three owners (Art, Bob, Carl) and five MRIs (123, 456, 789, 987, 654) that this definition treats as four entities (the boxes in the table).

**Table 1 – Example of how business entities are defined under group definition #1 (RFA)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Owner | | |  |
| MRI | Art | Bob | Carl |  |
| 123 | X | X | <= Entity 1 |  |
| 456 | X | X |  |
| 789 | X |  | X | <= Entity 2 |
| 987 |  | X | X | <= Entity 3 |
| 654 |  |  | X | <= Entity 4 |

Group Definition #2 Level (Interest Level – this is what was used in the groundfish reports):

Group definition #2 is the broadest definition of all used in this analysis. It is referred to as the interest level definition because it combines MRIs based on any possible “interest” (in a business sense) one owner might have with another owner. That is, if Art owns a vessel with Bob and Bob owns another vessel with Carl, all vessels are considered to be one business entity. The example below in Table 2 has the same owners and MRI relationships as in Table 1 but shows how all MRIs fall into one business entity under this definition. Note that it is not necessary for Bob to co-own MRI #987 with Carl to make this one business entity. If Carl owned MRI #987 himself, all of these MRIs would still be linked together because Art and Carl would both be co-owners with Bob even though they are not co-owners with each other.

**Table 2 – Example of how business entities are defined under group definition #2 (interest)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Owner | | |  |
| MRI | Art | Bob | Carl |  |
| 123 | X | X |  |  |
| 456 | X | X |  |
| 789 | X |  | X | <= Entity 1 |
| 987 |  | X | X |  |
| 654 |  |  | X |  |

It is not a necessary condition for Bob to own MRI #987 in order for this to be one entity

**SAS Program for Group Definition #1 (RFA Level)**

This example is for 2011 and 2012 only. Years can be added including adding the pre-2010 SQL scripts. This file can be found at ~akitts/peter2/Eric\_paper.

/\*\*\*\*\* BEGIN SAS PROGRAM \*\*\*\*\*/

%macro loops;

%do cy = 2011 %to 2012;

/\*\* FIRST THING IS TO GO GET PERSON AND PERMIT DATA FROM PERMIT DATABASES \*\*/

proc sql;

connect to oracle (user=xxxx orapw=xxxxxx path='sole');

%put &sqlxmsg;

%put &sqlxrc;

create view persons\_2011

as select \*

from connection to oracle

(

SELECT business\_ids.ap\_year AS ap\_year,

bo.person\_id AS person\_id,

cast(business\_ids.vp\_num AS VARCHAR(255)) AS permit

FROM (

SELECT ap\_nums.ap\_year,

ap\_nums.vp\_num,

ap\_nums.min\_ap\_num,

(SELECT min(start\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS start\_date,

(SELECT max(end\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS end\_date,

vo.business\_id

FROM (

SELECT ap\_year, vp\_num, min(ap\_num) as min\_ap\_num

FROM PERMIT.vps\_fishery\_ner

GROUP BY ap\_year, vp\_num

) ap\_nums

LEFT JOIN PERMIT.VPS\_OWNER vo ON vo.ap\_num = ap\_nums.min\_ap\_num AND vo.rank = 1

) business\_ids

LEFT JOIN PERMIT.BUSINESS\_OWNER bo ON business\_ids.business\_id = bo.business\_id AND bo.start\_date <= business\_ids.end\_date

AND (bo.end\_date IS NULL OR bo.end\_date >= business\_ids.start\_date)

WHERE business\_ids.ap\_year = 2011

);

create view persons\_2012

as select \*

from connection to oracle

(

SELECT business\_ids.ap\_year AS ap\_year,

bo.person\_id AS person\_id,

cast(business\_ids.vp\_num AS VARCHAR(255)) AS permit

FROM (

SELECT ap\_nums.ap\_year,

ap\_nums.vp\_num,

ap\_nums.min\_ap\_num,

(SELECT min(start\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS start\_date,

(SELECT max(end\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS end\_date,

vo.business\_id

FROM (

SELECT ap\_year, vp\_num, min(ap\_num) as min\_ap\_num

FROM PERMIT.vps\_fishery\_ner

GROUP BY ap\_year, vp\_num

) ap\_nums

LEFT JOIN PERMIT.VPS\_OWNER vo ON vo.ap\_num = ap\_nums.min\_ap\_num AND vo.rank = 1

) business\_ids

LEFT JOIN PERMIT.BUSINESS\_OWNER bo ON business\_ids.business\_id = bo.business\_id AND bo.start\_date <= business\_ids.end\_date

AND (bo.end\_date IS NULL OR bo.end\_date >= business\_ids.start\_date)

WHERE business\_ids.ap\_year = 2012

);

disconnect from oracle;

%put &sqlxmsg;

%put &sqlxrc;

quit;

libname eric '~akitts/peter2/Eric\_paper';

data one; set persons\_&cy;

cy = ap\_year;

permit2 = permit;

tag = 1;

if person\_id = . then person\_id = permit2;

proc sort; by cy person\_id;

data two; set one;

proc sort; by permit2 person\_id;

/\*\* NEXT WE ARRAY OUT ALL THE PERSONS ASSOCIATED WITH EACH PERMIT \*\*/

proc transpose data=two out=wide prefix=owner;

by permit2;

var person\_id;

data wide\_1; set wide;

array own {20} own1-own20;

%do i = 1 %to 20;

own{&i} = owner&i;

%end;

data wide2 (drop = \_name\_ \_label\_); set wide\_1;

tag = 1;

proc sort; by owner1-owner20;

proc means noprint data = wide2; by owner1-owner20;

var tag;

output out = wide3 sum=n\_permits;

/\*\* ...AND GIVE A UNIQUE OWNER\_ID TO EVERY UNIQUE COMBINATION OF PERSONS ASSOCIATED WITH A PERMIT \*\*/

data wide4; set wide3;

owner\_id = \_N\_;

proc sort; by owner\_id;

proc transpose data = wide4 out=long prefix = person\_id;

by owner\_id;

var owner1-owner20;

data long2 (drop = person\_id1 \_name\_); set long;

person\_id = person\_id1;

proc sort; by person\_id;

data three; set two;

proc sort; by person\_id;

data four; merge long2 three; by person\_id;

proc sort; by permit2;

proc sql; create view five as select distinct(permit2), owner\_id from work.four;

/\*\*\* THIS IS THE FINAL OWNER\_ID PERMIT MATCHING FILE \*\*\*/

data eric.RFA\_groups\_GFprepost\_&cy (keep = permit owner\_id cy); set five;

if owner\_id = . then delete;

if permit2 = . then delete;

tag = 1;

permit = input(permit2, best32.);

cy=&cy;

proc sort; by permit;

/\*\*\*\*\*NOTE THAT WHEN I MERGED THIS WITH PERMIT FILES I STILL GOT SOME

PERMITS WITHOUT AN EXACT MATCH, SO NO OWNER\_ID ... I ASSUMED THESE

WERE INDIVIDUAL OPERATORS AND CREATED AN OWNER\_ID FOR THEM AS

OWNER\_ID = \_N\_ \* 100000 \*\*\*\*\*\*\*/

%end;

%mend;

%loops;

/\*\*\*\*\* END SAS PROGRAM \*\*\*\*\*/

**PHP Program for Group Definition #2 (Interest Level)**

The following example uses post-2009 data and does not limit the initial data draw to any particular fishery. This file can be found at nova:/var/www/html/akitts.

/\*\*\*\*\* BEGIN PHP PROGRAM \*\*\*\*\*/

<?php

require "settings.php";

set\_time\_limit(300);

$conn = oci\_connect($dbSettings['username'], $dbSettings["password"], $dbSettings["hostname"]);

$sqlPost2010 = "SELECT business\_ids.ap\_year AS ap\_year,

bo.person\_id AS person\_id,

cast(business\_ids.vp\_num AS VARCHAR(255)) AS permit

FROM (

SELECT ap\_nums.ap\_year,

ap\_nums.vp\_num,

ap\_nums.min\_ap\_num,

(SELECT min(start\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS start\_date,

(SELECT max(end\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS end\_date,

vo.business\_id

FROM (

SELECT ap\_year, vp\_num, min(ap\_num) as min\_ap\_num

FROM PERMIT.vps\_fishery\_ner

GROUP BY ap\_year, vp\_num

) ap\_nums

LEFT JOIN PERMIT.VPS\_OWNER vo ON vo.ap\_num = ap\_nums.min\_ap\_num AND vo.rank = 1

) business\_ids

LEFT JOIN PERMIT.BUSINESS\_OWNER bo ON business\_ids.business\_id = bo.business\_id AND bo.start\_date <= business\_ids.end\_date AND (bo.end\_date IS NULL OR bo.end\_date >= business\_ids.start\_dat

WHERE business\_ids.ap\_year >= 2010";

$stid = oci\_parse($conn, $sqlPost2010);

if (!oci\_execute($stid)) {

return;

}

// Make an array which lists which permits belong to each person in a given year

$yearPersonPermits = array();

while ($row = oci\_fetch\_array($stid, OCI\_ASSOC+OCI\_RETURN\_NULLS)) {

$rowValues = array\_values($row);

$year = $rowValues[0];

$person\_id = $rowValues[1];

$permit = $rowValues[2];

$yearPersonPermits[$year][$person\_id][] = $permit;

}

// If we find two people that have a non-empty intersection of permits, remove the person with

// the higher person\_id, and assign all of their permits to the other person.

// Continue doing this until no more changes are found.

foreach ($yearPersonPermits as $year=>$personPermits) {

do {

$haschanges = false;

foreach ($personPermits as $person=>$permits) {

foreach ($personPermits as $person2=>$permits2) {

if ($person2 > $person && count(array\_intersect($permits, $permits2)) > 0) {

$haschanges = true;

unset($personPermits[$person2]);

$permits = $personPermits[$person] = array\_unique(array\_merge($permits, $permits2));

}

}

}

} while ($haschanges);

$yearPersonPermits[$year] = $personPermits;

}

$writeConn = null;

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$writeConn = oci\_connect($\_POST['username'], $\_POST['password'], $dbSettings['hostname']);

$table = $\_POST['table'];

echo "<p>Storing results in table ".$table."</p>";

oci\_execute(oci\_parse($writeConn, "autocommit off"));

oci\_execute(oci\_parse($writeConn, "CREATE TABLE $table (YEAR number, PERMIT number, GROUPID number)"));

oci\_execute(oci\_parse($writeConn, "CREATE UNIQUE INDEX ".$table."\_UQIDX ON $table (YEAR, PERMIT)"));

oci\_execute(oci\_parse($writeConn, "GRANT SELECT ON ".$table." TO ".$dbSettings['username']));

}

echo "<pre>";

echo "Year,Permit,GroupId\n";

// Assign the same "groupid" to all of the permits belonging to each person.

// The groupid is chosen to be the lowest valued permit number in the group

foreach ($yearPersonPermits as $year=>$personPermits) {

foreach ($personPermits as $permits) {

$groupid = min($permits);

foreach (array\_unique($permits) as $permit) {

echo $year,",",$permit,",",$groupid,"\n";

if (!is\_null($writeConn)) {

oci\_execute(oci\_parse($writeConn, "INSERT INTO $table (year, permit, groupid) values($year, $permit, $groupid)"));

}

}

}

}

if (!is\_null($writeConn)) {

oci\_execute(oci\_parse($writeConn, "COMMIT"));

}

?><form method="post">

<p>To write out these results to a database table, enter the following information:</p>

<ul>

<li><label>Table: <input type="text" name="table" value="permit\_groups"></label>

<li><label>Username: <input type="text" name="username" value="lshulman"></label>

<li><label>Password: <input type="password" name="password"></label>

</ul>

<input type="submit" value="Write Results to Table" />

</form>

/\*\*\*\*\* END PHP PROGRAM \*\*\*\*\*/

*How to run this PHP program:*

Point browser to <http://nova.wh.whoi.edu/akitts>. Then click on the file named [generateGroupIds\_post10\_all\_perms.php](http://nova.wh.whoi.edu/akitts/generateGroupIds_post10_all_perms.php). It will appear that your browser has stalled out (it will say it is connecting) but it hasn’t – the program is running. It may take up to a couple hours to finish. Just let it go…. When the program has finished, there will be 3 columns of data displayed on the screen. Scroll to the bottom. You will see:

Top of Form

To write out these results to a database table, enter the following information:

Table:  🡸 File name (can be changed…)

Username:  🡸 Change this to your username

Password:  🡸 Enter your password

Then click on the “Write Results to Table” button and the table will be saved in your table space which can be accessed thru SQLdeveloper.

Bottom of Form

The following example uses data from 1994 and beyond (pre and post 2010) and limits the initial data draw to the groundfish fishery. This file can be found at nova:/var/www/html/akitts.

/\*\*\*\*\* BEGIN PHP PROGRAM \*\*\*\*\*/

<?php

require "settings.php";

set\_time\_limit(300);

$conn = oci\_connect($dbSettings['username'], $dbSettings["password"], $dbSettings["hostname"]);

$sqlPre2010 = "SELECT min\_ap\_nums.ap\_year, ohp2.person\_id, min\_ap\_nums.permit

FROM (SELECT ap\_year, permit, min(ap\_num) as min\_ap\_num

FROM PERMIT.owner\_history\_pids

WHERE ap\_year < 2010

and person\_id is not null and permit is not null

AND ap\_num IN (SELECT ap\_num FROM PERMIT.vps\_fishery\_ner WHERE plan = 'MUL' AND cat in ('A','B','C','D','E','F','G','HA'))

GROUP BY ap\_year, permit) min\_ap\_nums

LEFT JOIN PERMIT.OWNER\_HISTORY\_PIDS ohp2 ON min\_ap\_nums.AP\_YEAR = ohp2.ap\_year AND min\_ap\_nums.permit = ohp2.permit AND min\_ap\_nums.min\_ap\_num = ohp2.ap\_num

";

$sqlPost2010 = "SELECT business\_ids.ap\_year AS ap\_year,

bo.person\_id AS person\_id,

cast(business\_ids.vp\_num AS VARCHAR(255)) AS permit

FROM (

SELECT ap\_nums.ap\_year,

ap\_nums.vp\_num,

ap\_nums.min\_ap\_num,

(SELECT min(start\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS start\_date,

(SELECT max(end\_date) FROM PERMIT.VPS\_FISHERY\_NER WHERE ap\_num = min\_ap\_num) AS end\_date,

vo.business\_id

FROM (

SELECT ap\_year, vp\_num, min(ap\_num) as min\_ap\_num

FROM PERMIT.vps\_fishery\_ner

WHERE plan = 'MUL' and cat in ('A','B','C','D','E','F','G','HA')

GROUP BY ap\_year, vp\_num

) ap\_nums

LEFT JOIN PERMIT.VPS\_OWNER vo ON vo.ap\_num = ap\_nums.min\_ap\_num AND vo.rank = 1

) business\_ids

LEFT JOIN PERMIT.BUSINESS\_OWNER bo ON business\_ids.business\_id = bo.business\_id AND bo.start\_date <= business\_ids.end\_date AND (bo.end\_date IS NULL OR bo.end\_date >= business\_ids.start\_dat

WHERE business\_ids.ap\_year >= 2010";

$stid = oci\_parse($conn, $sqlPre2010.' UNION '.$sqlPost2010);

if (!oci\_execute($stid)) {

return;

}

// Make an array which lists which permits belong to each person in a given year

$yearPersonPermits = array();

while ($row = oci\_fetch\_array($stid, OCI\_ASSOC+OCI\_RETURN\_NULLS)) {

$rowValues = array\_values($row);

$year = $rowValues[0];

$person\_id = $rowValues[1];

$permit = $rowValues[2];

$yearPersonPermits[$year][$person\_id][] = $permit;

}

// If we find two people that have a non-empty intersection of permits, remove the person with

// the higher person\_id, and assign all of their permits to the other person.

// Continue doing this until no more changes are found.

foreach ($yearPersonPermits as $year=>$personPermits) {

do {

$haschanges = false;

foreach ($personPermits as $person=>$permits) {

foreach ($personPermits as $person2=>$permits2) {

if ($person2 > $person && count(array\_intersect($permits, $permits2)) > 0) {

$haschanges = true;

unset($personPermits[$person2]);

$permits = $personPermits[$person] = array\_unique(array\_merge($permits, $permits2));

}

}

}

} while ($haschanges);

$yearPersonPermits[$year] = $personPermits;

}

$writeConn = null;

if ($\_SERVER['REQUEST\_METHOD'] == 'POST') {

$writeConn = oci\_connect($\_POST['username'], $\_POST['password'], $dbSettings['hostname']);

$table = $\_POST['table'];

echo "<p>Storing results in table ".$table."</p>";

oci\_execute(oci\_parse($writeConn, "autocommit off"));

oci\_execute(oci\_parse($writeConn, "CREATE TABLE $table (YEAR number, PERMIT number, GROUPID number)"));

oci\_execute(oci\_parse($writeConn, "CREATE UNIQUE INDEX ".$table."\_UQIDX ON $table (YEAR, PERMIT)"));

oci\_execute(oci\_parse($writeConn, "GRANT SELECT ON ".$table." TO ".$dbSettings['username']));

}

echo "<pre>";

echo "Year,Permit,GroupId\n";

// Assign the same "groupid" to all of the permits belonging to each person.

// The groupid is chosen to be the lowest valued permit number in the group

foreach ($yearPersonPermits as $year=>$personPermits) {

foreach ($personPermits as $permits) {

$groupid = min($permits);

foreach (array\_unique($permits) as $permit) {

echo $year,",",$permit,",",$groupid,"\n";

if (!is\_null($writeConn)) {

oci\_execute(oci\_parse($writeConn, "INSERT INTO $table (year, permit, groupid) values($year, $permit, $groupid)"));

}

}

}

}

if (!is\_null($writeConn)) {

oci\_execute(oci\_parse($writeConn, "COMMIT"));

}

?><form method="post">

<p>To write out these results to a database table, enter the following information:</p>

<ul>

<li><label>Table: <input type="text" name="table" value="permit\_groups"></label>

<li><label>Username: <input type="text" name="username" value="lshulman"></label>

<li><label>Password: <input type="password" name="password"></label>

</ul>

<input type="submit" value="Write Results to Table" />

</form>

/\*\*\*\*\* END PHP PROGRAM \*\*\*\*\*/

*How to run this PHP program:*

Point browser to <http://nova.wh.whoi.edu/akitts>. Then click on the file named [generateGroupIds.php](http://nova.wh.whoi.edu/akitts/generateGroupIds_post10_all_perms.php). It will appear that your browser has stalled out (it will say it is connecting) but it hasn’t – the program is running. It may take up to a couple hours to finish. Just let it go…. When the program has finished, there will be 3 columns of data displayed on the screen. Scroll to the bottom and follow instructions above.