



[About Us](#) |
 [Foundation](#) |
 [Operations](#) |
 [Regulatory](#) |
 [Decision Support](#) |
 [Directory](#) |
 [Publications](#)

30 January 2006



[Click to return to Database Documentation](#)

Task Description: Archival of Data and Database Preparation

Task: Copy data into 'historical' tables for permanent storage and prepare tables for next trapping season.

Rationale: To maintain a permanent copy of the trap data and to clear the tables for a new set of data.

Time line: Annually, usually in early spring prior to trapping season

Brief Process Description:

- Determine the maximum value in the ID columns of the GM_HISTORICAL table and the GM_HIST_INSPECT table. Create a sequence from which to select the ID value when inserting rows into these tables. The sequence may need to be dropped and re-created after copying data into the GM_HISTORICAL table and before copying data into the GM_HIST_INSPECT table.
- Copy rows from the data tables and snapshots (PLACEMENT, MI_PLACEMENT, MI_PLACEMENT_CTY, OMITTED_SITES, and MI_OMITS, and MI_OMITS_CTY) into the GM_HISTORICAL table.
- Copy rows from the data tables and snapshots (INSPECTION, MI_INSPECTION, and MI_INSPECTION_CTY) tables into the GM_HIST_INSPECT table.
- Delete all rows from the following tables in preparation for receiving the next season's data: PLACEMENT, MI_PLACEMENT, MI_PLACEMENT_CTY, OMITTED_SITES, MI_OMITS, MI_OMITS_CTY, INSPECTION, MI_INSPECTION, MI_INSPECTION_CTY, PEOPLE, BID_UNIT, SITE_LOCATIONS, SITE_LOCATIONS_CTY
- Insert new data into the SITE_LOCATIONS and SITE_LOCATIONS_CTY tables using SQL*Loader.

Processes:

Archive Data

The trap data are copied into tables where it will be permanently stored. The PLACEMENT and OMITTED_SITES data will be copied to the GM_HISTORICAL table, and the INSPECTION data will be copied to the GM_HIST_INSPECT table.

BEFORE you copy the data, make a FULL EXPORT of the database and run a FULL OFFLINE DATABASE BACKUP to removable storage media (tape/CD/DVD). (Refer to the task sheet on database export and offline backup for detailed instructions.) Keep this backup as a permanent copy of the database at that point in time. It is rare, but there may be a need to import objects from the export file at a later date. For instance, you would be able to retrieve the people table if someone needed information about one or more trappers. This request has been made once or twice in the past.

Copy Trap Placement/Omitted Sites Data:

Copy all rows from the trap placement and omitted sites tables at Virginia Tech and also from the snapshots that contain trap placement and omitted sites data from MSU (both quad-based and county-based data) into the GM_HISTORICAL table. The tables to copy are: OMITTED_SITES, PLACEMENT, MI_OMITS, MI_PLACEMENT, MI_OMITS_CTY, MI_PLACEMENT_CTY.

The ID value is unique for every row in the GM_HISTORICAL table; it is selected from a sequence as each row is inserted into the table. You will create the 'general_use' sequence beginning one digit greater than the maximum value in the ID column and then select from this sequence.

The current year (format 'YYYY') should be inserted into the YEAR column.

The INSERT statements are set up to insert into the columns in the order shown in the following description. A NULL value must be inserted as a placeholder for columns with no data.

GM_HISTORICAL Table Description:

ID	NUMBER(6)
AIPM_QUAD	VARCHAR2(3)
SITE	NUMBER(5)
UTM_EAST	NUMBER(6)
UTM_NORTH	NUMBER(7)
GRID	VARCHAR2(13)
PLACE_DATE	DATE
TOTAL_CATCH	NUMBER(5)
COUNTY	NUMBER(3)
STATE	VARCHAR2(2)
AGENCY	VARCHAR2(25)
POL_JURISDICTION	VARCHAR2(35)
TRAP_TYPE	VARCHAR2(11)
TRAPPER	VARCHAR2(3)
HABITAT	VARCHAR2(20)
GRIDNODE_UTME	NUMBER(6)
GRIDNODE_UTMN	NUMBER(7)

```

SENTINEL      VARCHAR2(1)
PROJECT       VARCHAR2(16)
USGS_CODE     VARCHAR2(8)
GENERATED_CATCH VARCHAR2(1)
YEAR          NUMBER(4)
GPS_DATA      VARCHAR2(1)
ENTRY_TYPE    VARCHAR2(8)
OMIT_REASON   VARCHAR2(30)
OUTSIDE_TARGET VARCHAR2(1)
DISTANCE_OUTSIDE NUMBER(7)
BLOCK_ID      NUMBER(2)
UTM_ZONE      NUMBER(2)
ELEVATION     NUMBER(5)

```

1. Determine maximum ID value in GM_HISTORICAL table

```
SELECT MAX(id) FROM gypsy.gm_historical;
```

2. Create sequence to be used for ID values (drop sequence if it already exists);

```
DROP SEQUENCE general_use; (if sequence already exists)
```

```
CREATE SEQUENCE general_use START WITH ; (start with value from the previous query +1)
```

3. Copy all rows from OMITTED_SITES table into GM_HISTORICAL:

```
INSERT INTO gypsy.gm_historical
```

```
SELECT general_use.nextval, NULL, O.site, O.utm_east, O.utm_north, O.grid, O.day, NULL, O.county, O.state, O.agency,
NULL, 'OMIT', L.initials, NULL, O.utm_east, O.utm_north, NULL, O.project, O.quad, NULL, 2006, O.gps_data, O.entry_type,
O.why, NULL, NULL, NULL, O.utm_zone, O.elevation
```

```
FROM gypsy.omitted_sites O, gypsy.people L
```

```
WHERE (O.state = L.state AND O.trapper = L.id);
```

4. Copy all rows from PLACEMENT table into GM_HISTORICAL:

```
INSERT INTO gypsy.gm_historical
```

```
SELECT general_use.nextval, NULL, P.site, P.utm_east, P.utm_north, P.grid, P.day, P.total_catch, P.county, P.state, P.agency,
NULL, P.trap_type, L.initials, NULL, P.gridnode_utme, P.gridnode_utm, P.sentinel, P.project, P.quad, NULL, 2006, P.gps_data,
P.entry_type, NULL, P.outside_target, P.distance_outside, NULL, P.utm_zone, P.elevation
```

```
FROM gypsy.placement P, gypsy.people L
```

```
WHERE (P.state = L.state AND P.trapper = L.id);
```

5. Copy all rows from MI_OMITS table into GM_HISTORICAL:

```
INSERT INTO gypsy.gm_historical
```

```
SELECT general_use.nextval, NULL, O.site, O.utm_east, O.utm_north, O.grid, O.day, NULL, O.county, O.state, O.agency,
NULL, 'OMIT', L.initials, NULL, O.utm_east, O.utm_north, NULL, O.project, O.quad, NULL, 2006, O.gps_data, O.entry_type,
O.why, NULL, NULL, NULL, O.utm_zone, O.elevation
```

```
FROM gypsy.mi_omits O, gypsy.mi_people L
```

```
WHERE (O.state = L.state AND O.trapper = L.id);
```

6. Copy all rows from MI_PLACEMENT table into GM_HISTORICAL:

```
INSERT INTO gypsy.gm_historical
```

```
SELECT general_use.nextval, NULL, P.site, P.utm_east, P.utm_north, P.grid, P.day, P.total_catch, P.county, P.state, P.agency,
NULL, P.trap_type, L.initials, NULL, P.gridnode_utme, P.gridnode_utm, P.sentinel, P.project, P.quad, NULL, 2006, P.gps_data,
P.entry_type, NULL, P.outside_target, P.distance_outside, NULL, P.utm_zone, P.elevation
```

```
FROM gypsy.mi_placement P, gypsy.mi_people L
```

```
WHERE (P.state = L.state AND P.trapper = L.id);
```

7. Copy all rows from MI_OMITS_CTY table into GM_HISTORICAL:

```
INSERT INTO gypsy.gm_historical
```

```
SELECT general_use.nextval, NULL, O.site, O.utm_east, O.utm_north, O.grid, O.day, NULL, O.county, O.state, O.agency,
NULL, 'OMIT', L.initials, NULL, O.utm_east, O.utm_north, NULL, O.project, O.quad, NULL, 2006, O.gps_data, O.entry_type,
O.why, NULL, NULL, O.block_id, O.utm_zone, O.elevation
```

```
FROM gypsy.mi_omits_cty O, gypsy.mi_people L
```

```
WHERE (O.state = L.state AND O.trapper = L.id);
```

8. Copy all rows from MI_PLACEMENT_CTY table into GM_HISTORICAL:

```
INSERT INTO gypsy.gm_historical
```

```
SELECT general_use.nextval, NULL, P.site, P.utm_east, P.utm_north, P.grid, P.day, P.total_catch, P.county, P.state, P.agency,
NULL, P.trap_type, L.initials, NULL, P.gridnode_utme, P.gridnode_utm, P.sentinel, P.project, P.quad, NULL, 2006, P.gps_data,
P.entry_type, NULL, P.outside_target, P.distance_outside, P.block_id, P.utm_zone, P.elevation
```

```
FROM gypsy.mi_placement_cty P, gypsy.mi_people L
```

```
WHERE (P.state = L.state AND P.trapper = L.id);
```

Copy Trap Inspection Data:

Copy all rows from the trap inspection tables at Virginia Tech and also from the snapshots that contain trap inspection data from MSU (both quad-based and county-based data) into the GM_HIST_INSPECT table. The tables to copy are: INSPECTION, MI_INSPECTION, MI_INSPECTION_CTY.

The ID value is unique for every row in the GM_HIST_INSPECT table; it is selected from a sequence as each row is inserted into the table. You will create the 'general_use' sequence beginning one digit greater than the maximum value in the ID column and then select from this sequence.

The current year (format 'YYYY') should be inserted into the YEAR column.

Since the inspection tables do not include a STATE column, we will include a subquery to the PLACEMENT table to determine which inspection rows belong to which state. The appropriate value will be included in the STATE column as each row is inserted into the GM_HIST_INSPECT table.

The INSERT statements are set up to insert into the columns in the order shown in the following description. A NULL value must be inserted as a placeholder for columns in which there is no data.

GM_HIST_INSPECT Table Description:

QUAD	VARCHAR2(8)
SITE	NUMBER(5)
DAY	DATE
VISIT	VARCHAR2(9)
CONDITION	VARCHAR2(12)
CATCH	NUMBER(4)
TRAPPER	VARCHAR2(3)
ID	NUMBER(6)
DEPTH	NUMBER(2)
FIELD_CHECK	VARCHAR2(1)
GPS_DATA	VARCHAR2(1)
QC_FAIL	NUMBER(2)
UTM_EAST	NUMBER(6)
UTM_NORTH	NUMBER(7)
ENTRY_TYPE	VARCHAR2(8)
YEAR	NUMBER(4)
STATE	VARCHAR2(2)
COUNTY	NUMBER(3)
BLOCK_ID	NUMBER(2)
UTM_ZONE	NUMBER(2)
ELEVATION	NUMBER(5)

1. Determine maximum ID value in GM_HIST_INSPECT table

```
SELECT MAX(id) FROM gypsy.gm_hist_inspect;
```

2. Create sequence to be used for ID values (drop sequence if it already exists);

```
DROP SEQUENCE general_use; (if sequence already exists)
```

```
CREATE SEQUENCE general_use START WITH ; (start with value from the previous query +1)
```

3. Copy all rows from INSPECTION table into GM_HIST_INSPECT:

```
INSERT INTO gypsy.gm_hist_inspect
```

```
SELECT A.quad, A.site, A.day, A.visit, A.condition, A.catch, L.initials, general_use.nextval, NULL, A.field_check, A.gps_data,
A.qc_fail, A.utm_east, A.utm_north, A.entry_type, 2006, 'KY', NULL, NULL, A.utm_zone, A.elevation
```

```
FROM gypsy.inspection A, gypsy.people L
```

```
WHERE (A.trapper = L.id) AND
```

```
(EXISTS (SELECT * FROM gypsy.placement P WHERE (P.state = 'KY') AND
```

```
(A.quad = P.quad AND A.site = P.site AND A.state = P.state)));
```

Continue to insert records for each state by changing the state abbreviation in the sub-query and re-running the INSERT statement. Also be sure to change the state value appropriately in the INSERT statement. Insert rows for each of the following states: KY, NC, OH, VA, WV.

4. Copy all rows from MI_INSPECTION table into GM_HIST_INSPECT:

```
INSERT INTO gypsy.gm_hist_inspect
```

```
SELECT A.quad, A.site, A.day, A.visit, A.condition, A.catch, L.initials, general_use.nextval, NULL, A.field_check, A.gps_data,
A.qc_fail, A.utm_east, A.utm_north, A.entry_type, 2006, 'IN', NULL, NULL, A.utm_zone, A.elevation
```

```
FROM gypsy.mi_inspection A, gypsy.mi_people L
```

```
WHERE (A.trapper = L.id) AND
```

```
(EXISTS (SELECT * FROM gypsy.mi_placement P WHERE (P.state = 'IN') AND
```

```
(A.quad = P.quad AND A.site = P.site AND A.state = P.state)));
```

Insert rows for each of the following states: IN, MI, MN.

5. Copy all rows from MI_INSPECTION_CTY table into GM_HIST_INSPECT:

```
INSERT INTO gypsy.gm_hist_inspect

SELECT A.quad, A.site, A.day, A.visit, A.condition, A.catch, L.initials, general_use.nextval, NULL, A.field_check, A.gps_data,
A.qc_fail, A.utm_east, A.utm_north, A.entry_type, 2006, 'IL', A.county, A.block_id, A.elevation

FROM gypsy.mi_inspection_cty A, gypsy.mi_people L

WHERE (A.trapper = L.id) AND

(EXISTS (SELECT * FROM gypsy.mi_placement_cty P WHERE (P.state = 'IL') AND

(A.state = P.state AND A.county = P.county AND A.block_id = P.block_id AND A.site = P.site)));
```

Repeat for Wisconsin.

CLEAR TABLES FOR NEW SURVEY DATA

Use the TRUNCATE command to delete all rows from each table. Foreign key constraints must be disabled before rows can be deleted. Re-enable constraints after deleting rows. The following tables need to be cleared:

```
BID_UNIT
FIRST_COORDINATES
FIRST_COORDINATES_CTY (MSU only)
INSPECTION
INSPECTION_CTY (MSU only)
OMITTED_SITES
OMITTED_SITES_CTY (MSU only)
PEOPLE
PLACEMENT
PLACEMENT_CTY (MSU only)
SITE_LOCATIONS
SITE_LOCATIONS_CTY
TEST_INSPECTION
TEST_INSPECTION_CTY (MSU only)
TEST_PLACEMENT
TEST_PLACEMENT_CTY (MSU only)
```

Quad-based Tables (at MSU and VT):

Log in to SQL*Plus as GYPSY and execute the following commands:

```
TRUNCATE TABLE site_locations;

TRUNCATE TABLE omitted_sites;

TRUNCATE TABLE inspection;

TRUNCATE TABLE errors;

TRUNCATE TABLE first_coordinates;

TRUNCATE TABLE test_inspection;

TRUNCATE TABLE test_placement;

ALTER TABLE inspection DISABLE CONSTRAINT fk_inspect_qdsitest;

TRUNCATE TABLE placement;

ALTER TABLE inspection ENABLE CONSTRAINT fk_inspect_quad_site;

ALTER TABLE site_locations DISABLE CONSTRAINT fk_siteloc_bidstate;

TRUNCATE TABLE bid_unit;

ALTER TABLE site_locations ENABLE CONSTRAINT fk_siteloc_bidstate;

ALTER TABLE people      DISABLE CONSTRAINT fk_people_superid;

ALTER TABLE placement   DISABLE CONSTRAINT fk_placement_trapper;

ALTER TABLE placement_cty DISABLE CONSTRAINT fk_place_cty_trapper;

ALTER TABLE inspection   DISABLE CONSTRAINT fk_inspect_trapper;

ALTER TABLE inspection_cty DISABLE CONSTRAINT fk_insp_cty_trapper;

ALTER TABLE bid_unit     DISABLE CONSTRAINT fk_bidunit_contractor;

ALTER TABLE omitted_sites DISABLE CONSTRAINT fk_omitted_trapper;

ALTER TABLE omitted_sites_cty  DISABLE CONSTRAINT fk_omit_cty_trapper;

ALTER TABLE site_locations  DISABLE CONSTRAINT fk_siteloc_trapper;

ALTER TABLE site_locations  DISABLE CONSTRAINT fk_siteloc_supervisor;

ALTER TABLE site_locations_cty DISABLE CONSTRAINT fk_siteloc_cty_trapper;

ALTER TABLE site_locations_cty  DISABLE CONSTRAINT fk_siteloc_cty_supervisor;
```

```

TRUNCATE TABLE people;

ALTER TABLE people      ENABLE CONSTRAINT fk_people_superid;

ALTER TABLE placement   ENABLE CONSTRAINT fk_placement_trapper;

ALTER TABLE placement_cty ENABLE CONSTRAINT fk_place_cty_trapper;

ALTER TABLE inspection   ENABLE CONSTRAINT fk_inspect_trapper;

ALTER TABLE inspection_cty ENABLE CONSTRAINT fk_insp_cty_trapper;

ALTER TABLE bid_unit     ENABLE CONSTRAINT fk_bidunit_contractor;

ALTER TABLE omitted_sites ENABLE CONSTRAINT fk_omitted_trapper;

ALTER TABLE omitted_sites_cty  ENABLE CONSTRAINT fk_omit_cty_trapper;

ALTER TABLE site_locations  ENABLE CONSTRAINT fk_siteloc_trapper;

ALTER TABLE site_locations  ENABLE CONSTRAINT fk_siteloc_supervisor;

ALTER TABLE site_locations_cty  ENABLE CONSTRAINT fk_siteloc_cty_trapper;

ALTER TABLE site_locations_cty  ENABLE CONSTRAINT fk_siteloc_cty_supervisor;

```

NOTE: not all constraints on placement_cty and inspect_cty are in place at VT. To house these functionalities at VT, these need to be copied from MSU and enabled. (fk_place_cty_trapper, fk_insp_cty_trapper, fk_omit_cty_trapper)

County-based Tables (at VT):

```
TRUNCATE TABLE site_locations_cty;
```

County-based Tables (at MSU):

```

TRUNCATE TABLE site_locations_cty;

TRUNCATE TABLE omitted_sites_cty;

TRUNCATE TABLE inspection_cty;

TRUNCATE TABLE first_coordinates_cty;

TRUNCATE TABLE test_inspection_cty;

TRUNCATE TABLE test_placement_cty;

ALTER TABLE inspection_cty DISABLE CONSTRAINT fk_inspect_cty_blk_site;

TRUNCATE TABLE placement_cty;

ALTER TABLE inspection_cty ENABLE CONSTRAINT fk_inspect_cty_blk_site;

```

LOAD DATA INTO REFERENCE TABLES

Before trap data can be loaded into the database, reference data must be inserted into the following tables: PEOPLE, SITE_LOCATIONS, SITE_LOCATIONS_CTY. If an agency assigns trapper territories (a.k.a. bid units) and would like reports organized in this way, then the BID_UNIT table will also need to be populated for that agency. The PEOPLE and BID_UNIT tables are loaded just before trapping begins since agencies usually do not have personnel hired until very close to the start of the trapping season. There is a referential integrity constraint between the CONTRACTOR column in the BID_UNIT table and the ID column in the PEOPLE table, so the trapper data must be inserted into the PEOPLE table before rows can be entered into the BID_UNIT table.

Load Data Into SITE_LOCATIONS and SITE_LOCATIONS_CTY Tables:

The predetermined sites are loaded into the SITE_LOCATIONS and SITE_LOCATIONS_CTY tables. SQL*Loader is used to load all of the predetermined sites into the tables at Virginia Tech. SELECT statements can then be run at MSU to populate the SITE_LOCATIONS and SITE_LOCATIONS_CTY tables there with the sites for IL, IN, MI, MN and WI.

The control files for SQL*Loader are located on mothsbane: **D:\data_load\sites\yyyy** (where yyyy = year of trapping season)

Comma-separated files will be written from the GIS (currently by Mannin Dodd) which can be loaded into the SITE_LOCATIONS and SITE_LOCATIONS_CTY tables. Quad-based sites will be written separately from the county-based sites so that each type can be loaded into the appropriate table. Mannin will most likely send the 'sites' files via email. The files should be copied to the directory on mothsbane where the control files are located:

D:\data_load\sites\yyyy (where yyyy = year of trapping season).

SITE_LOCATIONS

The control file 'site_locations_2006.ctl' is set up to load data from the 'sites' file into the SITE_LOCATIONS table in the following order:

```

UTM_EAST
UTM_NORTH
STATE
QUAD
AGENCY
COUNTY
GRID
PROJECT
TRAP_TYPE

```

BID_UNIT
SITE
UTM_ZONE

- Scroll through the 'sites' file to be sure the columns are in the same order as shown here. Check the values in the following columns to be sure they match the values in the DECODE statements for the following columns:

PROJECT – monit2, monit1, action, state
TRAP_TYPE – MC, Delta

- Check to be sure the GRID column is in the following format: 500M, 1K, 2K, 3K, 5K, etc.
- The QUAD column MUST contain the USGS code (e.g., 37086-B1). Quad names and abbreviations are not allowed in this column.
- The ID is assigned by Oracle as each row is loaded into the database beginning with one greater than the maximum ID value in the table.
- **Change** on the **third line (INFILE parameter)** the name of the file to be loaded. The filename entered on the third line should be the name of the 'sites' file that will be loaded.

SITE_LOCATIONS_CTY

The control file 'site_locations_cty_06.ctf' is set up to load data from the 'sites' file into the SITE_LOCATIONS_CTY table in the following order:

UTM_EAST
UTM_NORTH
STATE
QUAD
AGENCY
COUNTY
GRID
PROJECT
TRAP_TYPE
BID_UNIT
SITE
BLOCK_ID
UTM_ZONE

- Check to be sure the GRID column is in the following format: 1MI, 1X2MI, 2MI, etc.
- The QUAD column MUST contain the USGS code (e.g., 37086-B1). Quad names and abbreviations are not allowed in this column.
- The ID is assigned by Oracle as each row is loaded into the database beginning with one greater than the maximum ID value in the table.
- **Change** on the **third line (INFILE parameter)** the name of the file to be loaded. The filename entered on the third line should be the name of the 'sites' file that will be loaded.

PEOPLE

Information about every trapper is stored in the PEOPLE table. A sequence generates the ID value as each record is entered into the PEOPLE table. This sequence, PEOPLE_SEQ, can be dropped and re-created before the new trapper information is entered into the table. It is important that the name remain unchanged if the sequence is re-created. A trigger executes and selects the next value from the sequence PEOPLE_SEQ when a row is inserted into the PEOPLE table. To re-create the sequence, log in to the database as gypsy and type the following:

```
DROP SEQUENCE PEOPLE_SEQ;

CREATE SEQUENCE PEOPLE_SEQ START WITH 1;
```

Information about trappers must be entered into the PEOPLE table before trap data can be entered. The following items MUST be entered for each trapper:

INITIALS
FIRST_NAME
LAST_NAME
STATE
AGENCY
SUPERVISOR

Other items such as address and telephone number are optional.

BID_UNIT

Agencies that assign trappers to specific territories (or bid units) will provide that information prior to the beginning of the trapping season. The trapper information must be entered into the PEOPLE table before rows can be inserted into the BID_UNIT table. The following items must be entered for each trapper territory:

ID (territory number)
CONTRACTOR (Trapper ID from the PEOPLE table)
STATE (abbreviation)

After all site and trapper information has been loaded, the database should be ready to accept new trap data.

