

## Listing 1: logisticscreate

```
DROP DATABASE IF EXISTS members;
CREATE DATABASE members;
SHOW DATABASES;
USE members;
SELECT DATABASE();
create table if not exists member_information (
    memberID bigint unsigned not null default 0,
    name      varchar(18) not null default 'no_name',
    country   varchar(15) not null default 'no_country',
    email     varchar(30) not null default 'not_known',
    primary key(memberID),
) engine=InnoDB;

create table if not exists node_information (
    nodeID bigint unsigned not null default 0,
    name      varchar(18) not null default 'no_name',
    country   varchar(15) not null default 'no_country',
    email     varchar(30) not null default 'not_known',
    primary key(nodeID),
) engine=InnoDB;

create table if not exists privileges_information (
    privilegeID int unsigned not null default 0,
    privilegename varchar(18) not null default 'none',
    primary key(privilegeID)
) engine=InnoDB;

/* Junction table: node=0 is the member's global authorization.*/

create table if not exists member_authorizations (
    constraint c1 primary key (memberID, nodeID, privilegeID),
    foreign key (memberID) references member_information (memberID),
    foreign key (nodeID) references node_information (nodeID),
    foreign key (privilegeID) references privileges_information (privilegeID),
) engine=InnoDB;

/* Create a trigger to update the global authorizations table whenever a
node update's its local authorizations. Implements the GLAD access
control system using its "strictly conservative" strategy:
Procedure SC-strategy (Figure 3 in Castano et al. (1997). */

delimiter //
create trigger update_global_authorizations
after update
on member_authorizations

/* Count the number of nodes. */

set @nmnodes = (select count(nodeID) from node_information);

/* Delete old global authorizations. */

delete from member_authorizations where nodeID = 0;

/* Select member privileges that were given to them by all nodes. */

declare global temporary table n (
```

```

memberID bigint unsigned not null default 0,
nodeID bigint unsigned not null default 0,
privilegeID int unsigned not null default 0) not logged;

insert into session.n
select memberID, privilegeID, count(m.memberID) as nmgivenpriv
from member_authorizations m
having nmgivenpriv = @nmnodes
group by memberID, privilegeID;

update n
set nodeID = 0;

/* Replace the old global authorizations table with these new ones. */

insert into member_authorizations
select * from n;

/* Send an email to all nodes warning them that the global authorizations
   table has changed. */

system powershell ./updateemail.ps1
end//
delimiter ;

/* ----- Insert data into database ----- */

/* use traffickers; */

load data infile 'c:/polbio/wtsdbfiles/member.dat'
replace into table member_information
fields terminated by '\t'
optionally enclosed by '"'
lines terminated by '\r\n' ignore 1 lines
(memberID,name,country,email)

load data infile 'c:/polbio/wtsdbfiles/node.dat'
replace into table node_information
fields terminated by '\t'
optionally enclosed by '"'
lines terminated by '\r\n' ignore 1 lines
(nodeID,name,country,email)

load data infile 'c:/polbio/wtsdbfiles/privilege.dat'
replace into table privileges_information
fields terminated by '\t'
optionally enclosed by '"'
lines terminated by '\r\n' ignore 1 lines
(privilegeID,privilegename)

select * from member_information;

```

## Listing 2: updateemail.ps1

```
#
# PowerShell script to send an alert email to every node warning them that
# the global authorizations table has changed.

# Before running this script, run the following PowerShell script to
# encrypt the logistic node administrator's password:

# Read-Host -AsSecureString | ConvertFrom-SecureString | Out-File -FilePath
# "C:\Users\logisticsrootname\Documents\PSCredential\credtext.txt"

# -----

$EmailFrom = "logistics_root's_email_address"

$ToRecipients = "node1_email_address",
"node2_email_address",
"node3_email_address"

$EmailSubject = "Wildlife_crime_confederation_alert"
$EmailBody = "Global_authorizations_have_changed."

$SMTPserver= "SMTP_server"

$EncryptedPasswordFile = "$home\Documents\PSCredential\credtext.txt"
$username="logisticsrootname"
$password = Get-Content -Path $EncryptedPasswordFile | ConvertTo-SecureString
$credential =
New-Object System.Management.Automation.PSCredential($username, $password)

Send-MailMessage -ErrorAction Stop -from "$EmailFrom" -to $ToRecipients
-subject "$EmailSubject" -body "$EmailBody" -SmtpServer "$SMTPserver"
-Attachments "$home\Documents\PSlogs\Error_Log.txt" -Priority "Normal"
-Credential $credential -Port 587 -UseSsl
```

### Listing 3: nodecreate

```
DROP DATABASE IF EXISTS traffickers;
CREATE DATABASE traffickers;
SHOW DATABASES;
USE traffickers;
SELECT DATABASE();

/* ----- Confederation member tables. ----- */

create table if not exists member_information (
    memberID bigint unsigned not null default 0,
    name      varchar(18) not null default 'no_name',
    country   varchar(15) not null default 'no_country',
    email     varchar(30) not null default 'not_known',
    primary key(memberID),
) engine=federated
connection= 'mysql://root@logistics:3306/members/member_information';

create table if not exists privileges_information (
    privilegeID int unsigned not null default 0,
    privilegename varchar(18) not null default 'none',
    primary key(privilegeID)
) engine=federated
connection= 'mysql://root@logistics:3306/members/privileges_authorizations';

/* Junction table: node=0 is the member's global authorization.*/

create table if not exists member_authorizations (
    constraint c1 primary key (memberID, nodeID, privilegeID),
    foreign key (memberID) references member_information (memberID),
    foreign key (nodeID) references node_information (nodeID),
    foreign key (privilegeID) references privileges_information (privilegeID),
) engine=federated
connection= 'mysql://root@logistics:3306/members/member_authorizations';

/* ----- Criminal intelligence tables. ----- */

create table if not exists players (
    playerID bigint unsigned not null default 0,
    name      varchar(18) not null default 'no_name',
    country   varchar(15) not null default 'no_country',
    primary key(name)
) engine=InnoDB;

create table if not exists links (
    from_name varchar(30) not null,
    foreign key (from_name) references players(name)
        on delete cascade,
    to_name varchar(30) not null,
    foreign key (to_name) references players(name)
        on delete cascade,
    evidence_type varchar(30) not null
) engine=InnoDB;

create table if not exists phones (
    owner varchar(30),
    foreign key (owner) references players(name)
        on delete cascade,
```

```

    phone_number int unsigned not null default 000000
) engine=InnoDB;

create table if not exists cars (
    owner varchar(30),
    foreign key (owner) references players(name)
        on delete cascade,
    reg_number varchar(10) not null default ''
) engine=InnoDB;

create table if not exists guns (
    owner varchar(30),
    foreign key (owner) references players(name)
        on delete cascade,
    registration_number varchar(10) not null default ''
) engine=InnoDB;

create table if not exists ids (
    id bigint unsigned not null primary key,
    taken tinyint default 0
);

/* Update ids table with random ID numbers. To assign a new, random, and
   unique id, simply grab a random open one from the IDs table. Do this
   by grabbing a set of ids and attempting to make the UPDATE with one of
   them. If the update fails, then some other process grabbed that ID.
   That's no biggie, just try another. */

delimiter //
CREATE PROCEDURE randomid()
BEGIN
    DECLARE n INT DEFAULT 0;
    SELECT COUNT(*) FROM players INTO n;
    SET @i=0;
    WHILE @i<n DO
        set @val = (SELECT id FROM ids WHERE taken=0 LIMIT 1);
        update players set playerId=@val where playerId=0 limit 1;
        UPDATE ids SET taken=1 WHERE id=@val AND taken=0;
        SET @i = @i + 1;
    END WHILE;
End//
delimiter ;

/* Any time we run out of IDs, we can generate new ones with this stored
   procedure. */

delimiter //
CREATE PROCEDURE create_ids (create_id_num INT)
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLSTATE '42000' BEGIN END;
    SET @i = 1;
    loop1: LOOP
        INSERT INTO ids (id) VALUES (FLOOR(1 + RAND() * POW(2,63)));
        SET @i = @i + 1;
        IF @i > create_id_num THEN
            LEAVE loop1;
        END IF;
    END LOOP loop1;
END//

```

```

delimiter ;

/* Create a trigger that ensures that there are more IDs created when the
   last one is taken. */

delimiter //
CREATE TRIGGER make_additional_ids AFTER UPDATE ON ids
FOR EACH ROW
BEGIN
IF (SELECT COUNT(*) FROM ids WHERE taken = 0) = 0 THEN
    CALL create_ids(1000);
END IF;
END//
delimiter ;
call create_ids(100);

show tables;

/* ----- Insert data into database ----- */

/* use traffickers; */

load data infile 'c:/polbio/wtsdbfiles/traffickers.dat'
replace into table players
fields terminated by '\t'
optionally enclosed by '"'
lines terminated by '\r\n' ignore 1 lines
(name, country);

select * from players;

load data infile 'c:/polbio/wtsdbfiles/obsrvidlinks.dat'
replace into table links
fields terminated by '\t'
optionally enclosed by '"'
lines terminated by '\r\n' ignore 1 lines
(from_name, to_name, evidence_type);

select * from member_information;

```

## Listing 4: memberupdate

```
/* Updates this node's local authorizations table that is stored on the
   logistics node as the "member_authorizations" table. Doing so triggers
   the logistics node to update the global authorizations table
   (node 0 of this table). See the file "logistics_glad.sql".

   Update the table by first deleting all authorizations and then
   adding only those authorized by this node. */

delete from member_authorizations
where memberID = m22 and nodeID = 3;

insert into member_authorizations(memberID, nodeID, privilegeID)
values (m22, 3, 1),
       (m22, 3, 4);
```

## Listing 5: globalupdate

```
/* Updates user privileges on a node. This script is to be run by the
node's root when he/she receives an alert email from the logistics
node that the global authorizations table has changed. */

/* Revoke data access privileges for all users. */

DELIMITER //
CREATE PROCEDURE revokeprivs_()
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE memberid varchar(100) DEFAULT "";

    DECLARE currevoke CURSOR FOR
        select memberID
        from member_information;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

    OPEN currevoke;
    Mainloop: LOOP
        FETCH currevoke INTO memberid;
        IF finished = 1 THEN
            LEAVE Mainloop;
        END IF;

        revoke SELECT, INSERT, UPDATE, DELETE, REFERENCES, ALTER on *.* from
            memberid;

    END LOOP Mainloop;
    CLOSE currevoke;
END//
DELIMITER ;

call revokeprivs_();
flush privileges;

/* Now, set each user's privileges according to the updated global
authorizations table. */

DELIMITER //
CREATE PROCEDURE giveprivs_()
BEGIN
    DECLARE finished INTEGER DEFAULT 0;
    DECLARE memberid varchar(100) DEFAULT "";
    DECLARE privname varchar(100) DEFAULT "";

    DECLARE curgive CURSOR FOR
        select g.memberID, p.privilegename
        from member_authorizations g, privileges_information p
        where g.nodeID = 0 and g.privilegeID = p.privilegeID;

    DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

    OPEN curgive;
    Mainloop: LOOP
        FETCH curgive INTO memberid, privname;
        IF finished = 1 THEN
```



```
        LEAVE Mainloop;
    END IF;

    grant privname on *.* to memberid;

END LOOP Mainloop;
CLOSE curgive;
END//
DELIMITER ;

call giveprivs_();
flush privileges;
```

## Listing 6: fedquery.ps1

```
# Submits a query to all confederation nodes.

param ($queryfilename)

# User credentials are the same for all nodes.

$user = 'username'
$pass = 'password'

# Define confederation node URLs.

$nodeURL = @(
    '123.0'
    '123.1'
)
$nmnodes = $nodeURL.length

# Names of generic tables used by nodes. Index 0 on the first dimension
# is the generic name.

$nmtables = 2
$tablename = New-Object 'string[,]' ($nmnodes + 1), $nmtables
$tablename[0,0] = 'players'
$tablename[1,0] = 'suspect'
$tablename[2,0] = 'person'

$tablename[0,1] = 'guns'
$tablename[1,1] = 'weapon'
$tablename[2,1] = 'firearm'

# Names of generic columns used by nodes.

$nmcolumns = 2
$columnname = New-Object 'string[,]' ($nmnodes + 1), $nmcolumns
$columnname[0,0] = 'firstname'
$columnname[1,0] = 'first'
$columnname[2,0] = 'givenname'

$columnname[0,1] = 'lastname'
$columnname[1,1] = 'last'
$columnname[2,1] = 'surname'

# -----

function Main {

    $query = get-content($queryfilename)
    $query

# Loop over nodes.

    for ($i=0; $i -lt $nodeURL.length; $i++) {

        # Replace generic table and column names with those used by this node.

        $nodequery = $query
```

```

# Substitute node table and column names for the generic ones.

$cnode = $i + 1
for ($j=0; $j -lt $nmtables; $j++) {
    $nodequery = $nodequery -replace $tablename[0,$j], '
                                $tablename[$cnode,$j]
}
for ($j=0; $j -lt $nmcolums; $j++) {
    $nodequery = $nodequery -replace $columnname[0,$j], '
                                $columnname[$cnode,$j]
}

# Look up this node's URL and submit the remote SQL query.

<# Start-Process "./mysql -h$($nodeURL[$i]) -u$($user) -p$($pass)" '
    -RedirectStandardInput $nodequery '
    -RedirectStandardOutput "./temp.txt" '
    -NoNewWindow -Wait
#>
# Write output to stdout.

# $Result = Get-Content "./temp.txt"
    $i
    $nodequery
}
}

. Main

```

## Listing 7: example\_query

```
/* Submit this query to the confederation's database through the
powershell program written for this purpose with the command:

    powershell ./fedquery.ps1 example_query.sql > query_result.txt
*/

use traffickers;

select p1.playerID , p2.playerID
  from players p1, players p2, links l
  where p1.name = l.from_name and p2.name = l.to_name;
```