# voltdb-task-and-migrate-demo

This demo showcases some VoltDB features:

* The [MIGRATE](https://docs.voltdb.com/UsingVoltDB/sqlref_migrate.php) command.
* [Directed Procedures](https://docs.voltdb.com/UsingVoltDB/SimpleDirectedProcs.php) and Scheduled [Tasks](https://docs.voltdb.com/UsingVoltDB/ddlref_createtask.php).

## Scenario

We are pretending to be a system for detecting rogue drones in central London. Various locations are forbidden to drones. We need to raise the alarm if a drone gets too close to one of these locations. We also need to raise an alarm if we stop getting reports for a drone. We are expecting 10’s of thousands of position reports per second.

## Schema

We have the following tables, views and export streams:

|  |  |  |
| --- | --- | --- |
| Name | Type | Purpose |
| Drones | Table | Master table for drones. Used for tracking when to declare it missing. |
| Important\_Locations | Table | Places in Central London we want to keep an eye on |
| drone\_locations | Table | Keeps last 10 position reports for each drone. Extra rows are MIGRATED to old\_drone\_locations\_tgt |
| missing\_drones | Export Stream | We add a record every time a drone stops sending us location information |
| location\_incursions | Export Stream | We add a record every time a drone gets too close to a location mentioned in Important\_locations |
| old\_drone\_locations\_tgt | Export Stream | Where old drone\_location records go. Defined in the DDL for DRONE\_LOCATIONS |
| drone\_activity | View | Shows how many drones have reported positions over the last few minutes |
| missing\_drone\_stats | View | Shows how many drones will be declared missing over the next few minutes if we don’t get a location report |
| missing\_drone\_maxdate | View | A single row that tells us whether we need to search for missing drones |
| latest\_drone\_activity | View | A summary showing the latest location report time for each drone |

## Procedures

The application has two stored procedures:

### ReportLocation

‘ReportLocation’ is fed mock data by the test client ([TaskMigrateDemoClient](https://github.com/srmadscience/voltdb-task-and-migrate-demo/blob/master/src/org/voltdb/tastmigratedemo/TaskMigrateDemoClient.java)), but in a real world deployment would be configured to read messages from a queue such as Kafka or Kinesis.

[ReportLocation](https://github.com/srmadscience/voltdb-task-and-migrate-demo/blob/master/src/taskmigratedemo/ReportLocation.java) takes a position report for a drone and updates the database. It also:

* Checks to see if the drone is too close to an important\_location.
* [MIGRATES](https://docs.voltdb.com/UsingVoltDB/sqlref_migrate.php) any extra drone\_location records

This is called repeatedly from the demo’s client program.

### FindStaleDroneReports

[FindStaleDroneReports](https://github.com/srmadscience/voltdb-task-and-migrate-demo/blob/master/src/taskmigratedemo/FindStaleDroneReports.java) finds any drones that have failed to report for too long a time period and writes a message to missing\_drone\_stats. It also updates the drone record to prevent duplicate reports. FindStaleDroneReports is run as a [TASK](https://docs.voltdb.com/UsingVoltDB/ddlref_createtask.php) every 250 milliseconds

## Installation and setup

### VoltDB

To obtain VoltDB see [here](https://www.voltdb.com/try-voltdb/).

### Dependencies

This project needs [voltdb-schemabuilder](https://github.com/srmadscience/voltdb-schemabuilder).

### Configure Export Streams

In order to fully observe the functionality of this demo you’ll need to get export up and running. This can be done in two ways:

#### Configure Streams from within the VoltDB GUI

This can be done by going to the Admin tab of the VoltDB GUI and creating entries like these:

![Image of streams being configured](data:text/html; charset=utf-8;base64,)

Image of streams being configured

Note that you’ll need to change ‘outdir’ to a directory that exists on your computer. You could also set this up to go to Kafka (for example) by selecting it from the ‘type’ dropdown.

#### Configure streams by shutting down the database and editing deployment.xml

An alternative approach is to shut down VoltDB and then add the following entries to the database’s [deployment.xml[(https://docs.voltdb.com/UsingVoltDB/AppxConfigFile.php) configuration file:

<export>  
 <configuration target="old\_drone\_locations\_tgt" enabled="true" type="file" exportconnectorclass="" threadpool="">  
 <property name="type">csv</property>  
 <property name="nonce">odlt</property>  
 <property name="outdir">/Users/drolfe/csv</property>  
 </configuration>  
 <configuration target="tgt\_missing\_drones" enabled="true" type="file" exportconnectorclass="" threadpool="">  
 <property name="type">csv</property>  
 <property name="nonce">md</property>  
 <property name="outdir">/Users/drolfe/csv</property>  
 </configuration>  
 <configuration target="location\_incursions\_tgt" enabled="true" type="file">  
 <property name="type">csv</property>  
 <property name="nonce">lit</property>  
 <property name="outdir">/Users/drolfe/csv</property>  
 </configuration>  
 </export>

When you restart the database the export configuration should look like the one above.

### Run the code

The client program is called TaskMigrateDemoClient and takes the following parameters:

|  |  |  |
| --- | --- | --- |
| Parameter | Purpose | Example |
| hostnames | comma delimited list of VoltDB hosts | 127.0.0.1 |
| tps | How many transactions per second to generate | 50000 |
| size | How many drones to track | 1000000 |
| seconds | How many seconds to run for | 180 |

An example would be:

TaskMigrateDemoClient 192.168.0.50,192.168.0.51 50000 10000000 1800

Note that you may have to try various combinations of tps and size to get interesting results. The higher tps is the more records get written to ‘drone\_locations’, and the higher the value of ‘size’ the more likely it is that locations will get old enough to be flagged as ‘missing’ and reported to ‘missing\_drones’.

When running it calls the procedure GetStatus every 100,000 iterations, so you can see aggregate information about what is going on.

The first time it runs it starts by creating all the objects needed:

2019-12-16 08:54:32:Parameters:[127.0.0.1, 30000, 10000000, 1800]  
2019-12-16 08:54:32:Logging into VoltDB  
2019-12-16 08:54:32:Connect to 127.0.0.1...  
2019-12-16 08:54:32:Creating JAR file in /var/folders/\_0/ps5gxckx3\_lf5vc9jr4gz3yc0000gp/T/voltdbSchema4228325282647337994/taskMigrateProcs.jar  
2019-12-16 08:54:32:processing /taskmigratedemo/ReportLocation.class  
2019-12-16 08:54:32:  
2019-12-16 08:54:32:processing /taskmigratedemo/FindStaleDroneReports.class  
2019-12-16 08:54:32:  
2019-12-16 08:54:32:Calling @UpdateClasses to load JAR file containing procedures  
2019-12-16 08:54:33:create table important\_locations (location\_name varchar(20) not null primary key ,location\_latlong GEOGRAPHY\_POINT not null ,location\_exclusion\_zone\_radius\_m integer not null);  
2019-12-16 08:54:33:  
2019-12-16 08:54:33:CREATE TABLE drones (drone\_id bigint not null primary key, declare\_missing\_date timestamp);  
2019-12-16 08:54:33:  
2019-12-16 08:54:33:PARTITION TABLE drones ON COLUMN drone\_id;  
2019-12-16 08:54:33:  
2019-12-16 08:54:33:CREATE INDEX drone\_idx1 ON drones (declare\_missing\_date);  
2019-12-16 08:54:33:  
2019-12-16 08:54:33:CREATE TABLE drone\_locations MIGRATE TO TARGET old\_drone\_locations\_tgt (drone\_id bigint not null ,event\_timestamp timestamp not null ,drone\_location GEOGRAPHY\_POINT not null ,drone\_speed\_mps integer not null ,location\_is\_stale integer ,primary key (drone\_id, event\_timestamp));  
2019-12-16 08:54:33:  
2019-12-16 08:54:34:PARTITION TABLE drone\_locations ON COLUMN drone\_id;  
2019-12-16 08:54:34:  
2019-12-16 08:54:34:CREATE INDEX drone\_location\_ts\_idx ON drone\_locations (drone\_id, event\_timestamp);  
2019-12-16 08:54:34:  
2019-12-16 08:54:34:CREATE STREAM missing\_drones PARTITION ON COLUMN drone\_id EXPORT TO TARGET tgt\_missing\_drones(drone\_id bigint not null ,event\_timestamp timestamp not null ,drone\_location GEOGRAPHY\_POINT not null ,drone\_speed\_mps integer not null ,location\_is\_stale integer);  
2019-12-16 08:54:34:  
2019-12-16 08:54:35:CREATE STREAM location\_incursions PARTITION ON COLUMN drone\_id EXPORT TO TARGET location\_incursions\_tgt (drone\_id bigint not null ,event\_timestamp timestamp not null ,drone\_location GEOGRAPHY\_POINT not null ,drone\_speed\_mps integer not null ,location\_name varchar(20) not null ,distance\_from\_metres bigint not null );  
2019-12-16 08:54:35:  
2019-12-16 08:54:35:CREATE VIEW drone\_activity AS SELECT TRUNCATE(MINUTE, event\_timestamp) activity\_minute, COUNT(\*) HOW\_MANY FROM drone\_locations GROUP BY TRUNCATE(MINUTE, event\_timestamp);  
2019-12-16 08:54:35:  
2019-12-16 08:54:35:CREATE VIEW missing\_drone\_stats AS SELECT TRUNCATE(MINUTE, declare\_missing\_date) declare\_missing\_date, COUNT(\*) HOW\_MANY FROM drones GROUP BY TRUNCATE(MINUTE, declare\_missing\_date);  
2019-12-16 08:54:35:  
2019-12-16 08:54:35:CREATE VIEW missing\_drone\_maxdate AS SELECT MAX(declare\_missing\_date) declare\_missing\_date FROM drones;  
2019-12-16 08:54:35:  
2019-12-16 08:54:36:CREATE VIEW latest\_drone\_activity AS SELECT drone\_id, max(event\_timestamp) event\_timestamp FROM drone\_locations GROUP BY drone\_id;  
2019-12-16 08:54:36:  
2019-12-16 08:54:36:CREATE INDEX lda\_index1 ON latest\_drone\_activity(event\_timestamp, drone\_id);   
2019-12-16 08:54:36:  
2019-12-16 08:54:36:CREATE PROCEDURE PARTITION ON TABLE drone\_locations COLUMN drone\_id FROM CLASS taskmigratedemo.ReportLocation;  
2019-12-16 08:54:36:  
2019-12-16 08:54:36:CREATE PROCEDURE DIRECTED FROM CLASS taskmigratedemo.FindStaleDroneReports;  
2019-12-16 08:54:36:  
2019-12-16 08:54:36:CREATE PROCEDURE GetDrone PARTITION ON TABLE drone\_locations COLUMN drone\_id AS SELECT \* FROM drone\_locations WHERE drone\_id = ? ORDER BY event\_timestamp DESC;  
2019-12-16 08:54:36:  
2019-12-16 08:54:37:CREATE PROCEDURE GetStatus AS BEGIN SELECT Activity\_minute last\_active, how\_many FROM drone\_activity ORDER BY activity\_minute; SELECT HOW\_MANY FROM missing\_drone\_stats WHERE DECLARE\_MISSING\_DATE IS NULL ORDER BY declare\_missing\_date; END;  
2019-12-16 08:54:37:  
2019-12-16 08:54:37:CREATE TASK findStaleDronesTask ON SCHEDULE DELAY 250 MILLISECONDS PROCEDURE FindStaleDroneReports ON ERROR LOG RUN ON PARTITIONS;  
2019-12-16 08:54:37:

It then sends ReportLocation messages to the server, at a rate of ‘tps’ per second. Every 100,000 events we call GetStatus, which show how many drones have been active over the last few minutes and how many appear to be missing:

2019-12-16 08:54:37:Starting test run at 30000 transactions per second for 1800 seconds  
2019-12-16 08:54:41:Transaction #100000  
2019-12-16 08:54:41:Drone Activity By Minute:  
2019-12-16 08:54:41:  
LAST\_ACTIVE HOW\_MANY   
--------------------------- ---------  
2019-12-16 08:54:00.000000 100000  
  
2019-12-16 08:54:41:Missing Drones By Minute:  
2019-12-16 08:54:41:  
HOW\_MANY   
---------  
 36225  
  
2019-12-16 08:54:45:Transaction #200000  
2019-12-16 08:54:45:Drone Activity By Minute:  
2019-12-16 08:54:45:  
LAST\_ACTIVE HOW\_MANY   
--------------------------- ---------  
2019-12-16 08:54:00.000000 200000  
  
2019-12-16 08:54:45:Missing Drones By Minute:  
2019-12-16 08:54:45:  
HOW\_MANY   
---------  
 69825

When running the ‘Db Monitor’ tab of the VoltDB GUI should look like this:

![db monitor](data:text/html; charset=utf-8;base64,)

db monitor

As the demo progresses the Export tab will show records being written:

![export_tab](data:text/html; charset=utf-8;base64,)

export\_tab

Meanwhile if you look at the directory you requested the file exporter to write to you’ll see files being written:

Davids-MacBook-Pro-7:csv drolfe$ pwd  
/Users/drolfe/csv  
Davids-MacBook-Pro-7:csv drolfe$ ls -altr  
total 6457160  
drwxr-xr-x+ 150 drolfe staff 4800 Dec 16 07:54 ..  
-rw-r--r-- 1 drolfe staff 77275 Dec 16 09:13 active-location\_incursions\_tgt-3165364684111183871-LOCATION\_INCURSIONS-20191216085435.csv  
drwxr-xr-x 5 drolfe staff 160 Dec 16 09:15 .  
-rw-r--r-- 1 drolfe staff 1834891267 Dec 16 09:22 active-tgt\_missing\_drones-3165364684111183871-MISSING\_DRONES-20191216085435.csv  
-rw-r--r-- 1 drolfe staff 1457170935 Dec 16 09:22 active-old\_drone\_locations\_tgt-3165364684111183871-DRONE\_LOCATIONS-20191216085435.csv

And a peek inside the LOCATION\_INCURSIONS file shows that someone has got too close to Buckingham Palace:

"3619671939874816","1199522942031414","57","0","0","1","15","2019-12-16 09:13:51.414","POINT (-0.1436013 51.5013606)","2","Buckingham Palace","934"  
"3619671939891200","1199522942031417","58","0","0","1","21","2019-12-16 09:13:51.417","POINT (-0.1436013 51.5013606)","3","Buckingham Palace","940"  
"3619671939907584","1199522942031422","59","0","0","1","34","2019-12-16 09:13:51.422","POINT (-0.1436013 51.5013606)","7","Buckingham Palace","958"  
"3619671939923968","1199522942031423","60","0","0","1","38","2019-12-16 09:13:51.423","POINT (-0.1436013 51.5013606)","3","Buckingham Palace","965"  
"3619671939940352","1199522942031424","61","0","0","1","40","2019-12-16 09:13:51.424","POINT (-0.1436013 51.5013606)","9","Buckingham Palace","969"  
"3619671939956736","1199522942031424","62","0","0","1","41","2019-12-16 09:13:51.424","POINT (-0.1436013 51.5013606)","5","Buckingham Palace","971"

## Conclusion

In this demo we’ve shown how you can use VoltDB to create a fast, scalable application that takes important, stateful decisions in real time. In addition to showing how we can can correlate the latest position of a device with its proximity to a location we also show how we can take *smart* decisions, such as reporting a device as missing once (and only once).