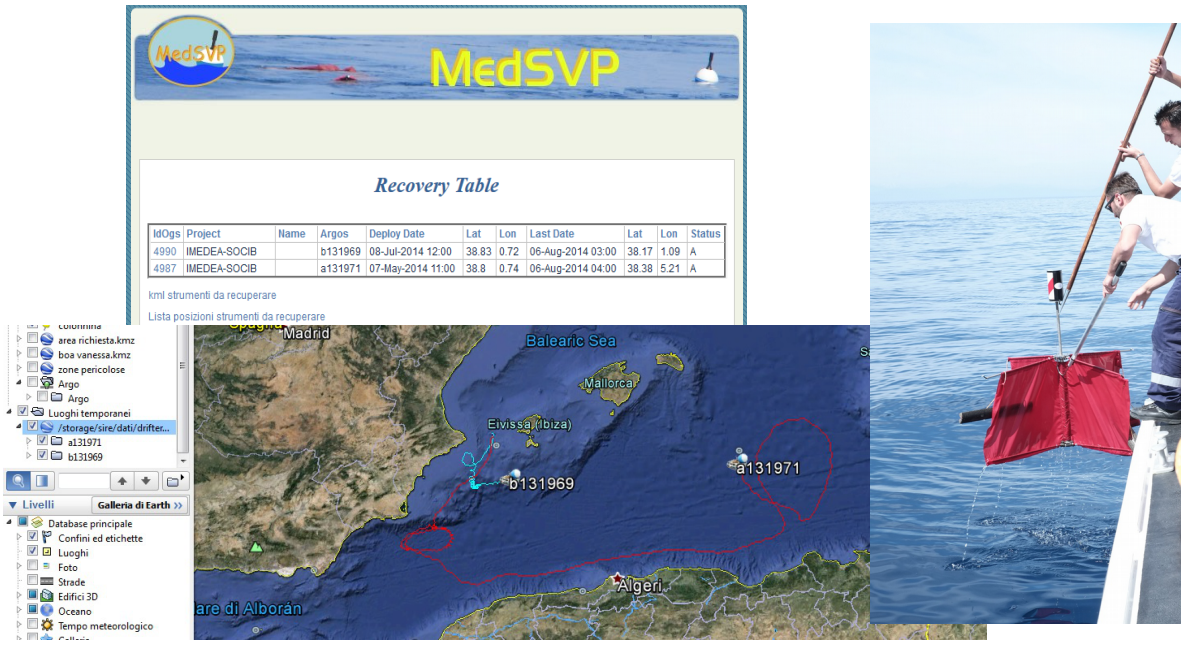


Prodotti web e notifiche a supporto delle operazioni di recupero dei drifter

A. BUSSANI e R. GERIN

Approved by: ……

Dr. Paola Del Negro

INDICE:

Abstract

Nei progetti di ricerca degli ultimi anni è stato impiegato un numero sempre maggiore di strumenti ed in particolare di drifter. Tali strumenti spesso devono essere recuperati e rimessi a mare il più rapidamente possibile. È pertanto fondamentale avere una visione del loro posizionamento in tempo reale (o semi reale) che sia completa, chiara e precisa. I programmi per la visualizzazione e la produzione delle pagine web standard possono risultare non efficaci in quanto troppo lenti. Vista l'immane quantità di drifter e script che essi amministrano, è impensabile farli girare con frequenza elevata. Pertanto, al fine di riuscire a produrre il più rapidamente possibile le elaborazioni necessarie al recupero di selezionati strumenti, sono stati creati:

- una pagina web apposita per il recupero della strumentazione scelta;
- dei file di testo ove sono indicati la posizione dello strumento ed eventuali altri dettagli;
- delle immagini per singolo strumento con la sua traiettoria;
- dei file kml (google earth) con i dati di traiettoria;
- una immagine riepilogativa con tutti gli strumenti da recuperare (e le relative traiettorie);
- un file kml riepilogativo con tutti gli strumenti da recuperare (e le relative traiettorie).

I programmi creati inoltre permettono la generazione delle immagini ricavandole dal dato decodificato (con gli spike, con/senza il filtraggio dei dati a terra), o dal dato editato.

I programmi creati (per il momento), non includono

- la spedizione di mail indicanti l'ultima posizione dei drifter scelti
- la spedizione di SMS con le ultime posizioni dei drifter scelti

Queste particolarità sono parte integrante di altre procedure che possono essere facilmente eseguite all'occorrenza.
Va infine sottolineato che gli script utili per il recupero di drifter possono essere usati anche per la generazione delle immagini (e file kml) delle pagine web relative ai progetti e/o alle immagini dei singoli deploy.
Il report va inteso come manuale di istruzioni da seguire durante le fasi di recupero dei drifter e descrive dettagliatamente il flusso di dati e gli script utili alla generazione delle pagine web e dei prodotti di supporto.

Script di partenza

Lo script che include tutte le procedure per l'esecuzione selettiva delle elaborazioni è il seguente:

```
/storage/sire/work/drifter/script/updateDataDrifter_recdb
```

Lo script è inserito all'interno del crontab con utenza drifter.

```
crontab -l
#*/30 * * * * /storage/sire/work/drifter/script/updateDataDrifter_recdb
>/dev/null
```

Scaricamento dati – pre processing

Per poter creare i prodotti utili al recupero, è necessario avere i dati aggiornati. Quindi è necessario innanzitutto scaricare dalle fonti opportune i nuovi dati.
Le fonti dalle quali possono essere recuperati sono:

- e-mail
- telnet (argos)
- ftp ogs
- ftp esterni
- siti web esterni (vedi caso Socib)
- SMS
- cartelle locali aggiornate in tempo reale (es rudics)

Nel caso dei drifter all'interno dello script:
`/storage/sire/work/drifter/script/PreProcessing-rec.sh`
è possibile inserire i diversi script specializzati per singoli strumenti (in modo da ridurne il loro numero e quindi ridurre il più possibile i tempi di scaricamento).

Come indicare gli strumenti da recuperare

Inizialmente si è pensato di elaborare esclusivamente gli strumenti attivi, ma con questa scelta (lasciando i parametri di recupero intatti), quando lo strumento veniva recuperato (quindi immessa la data di recupero nel campo end_life_time), la procedura di decoding lo poneva come drifter morto e l'esecuzione (per quel drifter) non avveniva.
Visto inoltre che lo script viene usato anche come rielaborazione forzata a posteriori (del recupero), si è pensato di lasciare attiva l'elaborazione anche per gli strumenti non attivi, fermo restando la loro elaborazione nel caso della presenza di dati diversi da zero nel campo dati_recupero_rt.

Campi database interessati

All'interno della tabella tbl_drifter il campo dati_recupero_rt, è quello che deve contenere un numero che identifica il/i tipi di elaborazione necessari per il recupero.

La quantità di elaborazioni dovrebbe essere equilibrata a seconda della quantità di strumenti da recuperare, rispetto al tempo relativo alla loro elaborazione (in funzione al numero di processori del server che esegue l'operazione).
T_{tot}=num_strum * $T_{elab\text{-}completa}$
Per ora non è stata presa in considerazione la parallelizzazione parziale della procedura, ma sarà necessario produrla in seguito.

Il valore da includere nel campo dipende dal tipo di elaborazione (elencate nella successiva tabella), è possibile sommare i vari valori per ottenere gli effetti cumulativi relativi:

Posizione	Valore	Descrizione
-	0	non produce nulla – vengono saltate tutte le elaborazioni (tutte le posizioni binarie sono 0)
0 oppure 32 (per numerazione SQL)	2^0=1	Elaborazione forzata completa (non solo gli aggiornamenti). Viene eseguita solo per parametri di "decodifica" matlab: correct, decod, editing, kriging, netcdf, argv. Non tutte le funzioni eseguono (per ora) l'analisi parziale.
1 oppure 31	2^1=2	correct
2 oppure 30	2^2=4	decod
3 oppure 29	2^3=8	Editing
4 oppure 28	2^4=16	Se settato produce l'editing lasciando i punti a terra → no_filtro_terra settato a 1
5 oppure 27	2^5=32	kriging
6 oppure 26	2^6=64	netcdf_new (per creazione netcdf) → myocean
7 oppure 25	2^7=128	argv
8 oppure 24	2^8=256	uso futuro
9 oppure 23	2^9=512	uso futuro
10 oppure 22	2^{10}=1024	immagini di deploy (singolo strumento) - matlab, (tutti non solo per alive)
11 oppure 21	2^{11}=2048	immagini di deploy (singolo strumento) - kml, (tutti non solo per alive)
12 oppure 20	2^{12}=4096	immagini di deploy (gruppo rec) – kml, (tutti non solo per alive)
13 oppure 19	2^{13}=8192	immagini di deploy (gruppo di deploy) – web_pages possibile esistenza di diversi progetti per singolo strumento
14 oppure 18	2^{14}=16384	immagini per l"intero progetto – country_www -
15 oppure 17	2^{15}=32768	Genera kml per 13 e 14-
16 oppure 16	2^{16}=65536	Genera png per 13 e 14-
17 oppure 15	2^{17}=131072	Crea immagini di gruppo solo quelli vivi se settato a 1. Per inserirli tutti (anche quelli morti) inserire il valore 0 in questa posizione. (Quindi modifica il comportamento delle posizioni 13, 14, 15, 16).
18 oppure 14	2^{18}=262144	1 in questo campo indica che solo i file .kml/.kmz, se questi devono essere generati, andranno creati partendo dai dati decodificati e non da quelli editati.
...	...	...
...	...	...
30 oppure	2^{30}=1073741824	uso futuro

In postgreqsl il valore massimo per una variabile integer è 2147483647.
Esattamente:

integer	4 bytes	From -2147483648 to +2147483647

Con tale valore massimo è possibile (escludendo i numeri negativi) includere valori fino a 2^30. offrendo quindi 31 (da 0 a 30) differenti tipi di elaborazioni diverse.

Se si vogliono più funzioni contemporaneamente è necessario sommare i valori corrispondenti, ad esempio 3 per avere sia i file txt che le immagini del singolo strumento.

Esempi di utilizzo

Dopo aver scelto:
- quali drifter si vuole recuperare
- che visualizzazioni/notifiche effettuare

è possibile inserire tale dato nel db.

Es:
voglio far si che tutti i drifter del progetto MEDESS, vengano elaborati e mostrati.
Le visualizzazioni che mi servono sono:
1 – elaborazione forzata
2 – correzione
4 – decodifica
128 – creazione degli argv
4096 – creazione del kml denominato gruppo_rec (in un singolo file google earth tutti i drifter del progetto Medess
Totale: 4231

Tale valore va quindi inserito nel db su ogni singolo drifter di Medess:
sql:

```
UPDATE tbl_drifter SET tbl_drifter.dati_recupero_rt = 4231
WHERE ((tbl_drifter.country_www="MEDESS"));
```

da shell (utente drifter)

```
 psql -t -d float -c $'UPDATE tbl_drifter SET dati_recupero_rt = 4231
WHERE (country_www = \'MEDESS\');'
UPDATE 17
```

In questo esempio 17 drifter sono stati aggiornati con il valore 4231
Per averne conferma, da shell (utente drifter):

```
psql -t -d float -c $'SELECT id_drifter, dati_recupero_rt FROM tbl_drifter WHERE
(country_www = \'MEDESS\');'
       4943 |             4231
       4933 |             4231
       4922 |             4231
       4939 |             4231
       4946 |             4231
       4947 |             4231
       4927 |             4231
```

```
       4928 |              4231
       4940 |              4231
       4945 |              4231
       4925 |              4231
       4913 |              4231
       4923 |              4231
       4924 |              4231
       4942 |              4231
       4914 |              4231
       4918 |              4231
```

per resettare le opzioni inserite basta porre a 0 il campo relativo:

```
psql -t -d float -c $'UPDATE tbl_drifter SET dati_recupero_rt = 0 WHERE
(country_www = \'MEDESS\');'
UPDATE 17
[drifter@oceano script]$  psql -t -d float -c $'SELECT id_drifter,
dati_recupero_rt FROM tbl_drifter WHERE (country_www = \'MEDESS\');'
       4943 |                 0
       4933 |                 0
       4922 |                 0
       4939 |                 0
       4946 |                 0
       4947 |                 0
       4927 |                 0
       4928 |                 0
       4940 |                 0
       4945 |                 0
       4925 |                 0
       4913 |                 0
       4923 |                 0
       4924 |                 0
       4942 |                 0
       4914 |                 0
       4918 |                 0
```

Come variare la frequenza delle elaborazioni per gli strumenti da recuperare

La frequenza dell'esecuzione è inserita all'interno del cron. Per i cambiamenti sul cron (utenza drifter) è necessario usare questo comando:

```
crontab -e
```

Questo caricherà l'editor di default (vi) e permetterà la variazione del numero di esecuzioni.
La riga da variare è la seguente:

```
*/30 * * * * /storage/sire/work/drifter/script/updateDataDrifter_recdb
>/dev/null
```

Il primo campo identifica ogni quanti minuti verrà eseguito lo script:

- inserendo ad es. 10 lo script verrà eseguito ad ogni ora quando scattano i 10 minuti
- inserendo 10,20 lo script verrà eseguito ad ogni ora quando scattano i 10 e i 20 minuti
- inserendo */10 lo script verrà eseguito ad ogni ora ogni 10 minuti

è necessario tenere conto di non aumentare troppo la frequenza che deve essere sempre inferiore al tempo di esecuzione dell'intero script, altrimenti la coda dell'esecuzione non può essere smaltita, provocando un blocco del sistema a causa della mancanza di risorse.

Man mano che il progetto avanza i dati si accumulano e lo script risulta più lento, bisogna tenerne conto e regolare la frequenza di conseguenza.

Script di elaborazione

All'interno di
`/storage/sire/work/drifter/script/updateDataDrifter_recdb`
è possibile trovare le singole chiamate agli script secondari e la creazione di quelli ad hoc.

L'esecuzione del codice matlab per il recupero avviene nello script:
MAIN_recdb.m

```
MATLABR='/usr/local/bin/matlab -nodisplay -nosplash -timing -r '
echo "$MATLAB <$MATLABDIR/MAIN_recdb.m" >> $STDOUT
$MATLABR "try; run '$MATLABDIR/MAIN_recdb.m'; catch; end; quit" >> $STDOUT
echo "--- fine MATLAB < MAIN_recdb.m " >> $STDOUT
```

L'esecuzione dello script matlab attraverso la shell è subordinata dall'utilizzo dei parametri -nodisplay, il parametro -r permette l'esecuzione del codice inserito di seguito.
Precedentemente, i vari script venivano eseguiti usando il redirect (<), tale procedura però risultava non funzionante per script maggiori di 512kb.

Di seguito il codice per la creazione della where in matlab: where_bit

```
%crea la sezione where di tipo sql, per rintracciare il bit numero (in) per
%il campo tbl_drifter.dati_recupero_rt
%la numerazione dei bit è i tipo big endian
%0001 risulta 0*2^3+0*2^2+0*2^1+1*2^0
%il numero in e' un numero a 32 bit, la posizione viene conteggiata in modo
%che nell'esempio precedente (0001) il bit a 1 si trovi nella posizione 0
function out=where_bit(in)
    position=33-
    str='((substring(cast(dati_recupero_rt as int4)::bit(32) from '
32 for 1)) = '1')';
end

sql where
substring(cast(dati_recupero_rt as int4)::bit(32) from 32 for 1)='1'
```

Di seguito il codice per la scelta della tipologia di dati (decodificati o editati)

```
decod="$(psql -t -d float -c $'SELECT DISTINCT substring(cast(dati_recupero_rt
as int4)::bit(32) from 14 for 1) as dec FROM tbl_drifter WHERE
dati_recupero_rt<>0;')"
dec_no_space=`echo $decod`

if [ -n "$dec_no_space" ]; then
if [ $dec_no_space -eq 1 ]; then
DATIELAB=$DATIELABD
else
DATIELAB=$DATIELABE
fi
else
DATIELAB=$DATIELABE
```

```
fi
echo "Dati usati per il kml: $DATIELAB"
```

Di seguito il codice per un singolo parametro

```
## --- 4096 --- ##
#sql per generazione google earth (kml) group
AND_WHERE=" ((substring(cast(dati_recupero_rt as int4)::bit(32) from 20 for 1))
= '1')"
#echo "AND_WHERE=$AND_WHERE"
echo "Generazione mappe_singoli bit posizione 20, valore 4096"  >> $STDOUT

$COMMONDIR/list_instr_mfile.pl Drifter "SELECT CASE WHEN imei is null THEN
(argo_mission || id_argo || '.mat')  ELSE (argo_mission || IMEI || '.mat') END
AS mat_drifter, tbl_type.icon, tbl_type.description, tbl_drifter.country_www,
tbl_type.id_type, tbl_drifter.wmo
 FROM tbl_drifter INNER JOIN tbl_type ON tbl_drifter.id_type = tbl_type.id_type
 WHERE $AND_WHERE;"  /tmp/drifter_kml_rec_group.m
"kml_generator(lista_matfile,type,'$DATIELAB','/storage/sire/dati/drifter/elab/k
ml/','drifter_kml_rec_group.kmz','group')" >> $STDOUT
$PAR $MATLABR "try; run '/tmp/drifter_kml_rec_group.m'; catch; end; quit" >>
$STDOUT
```

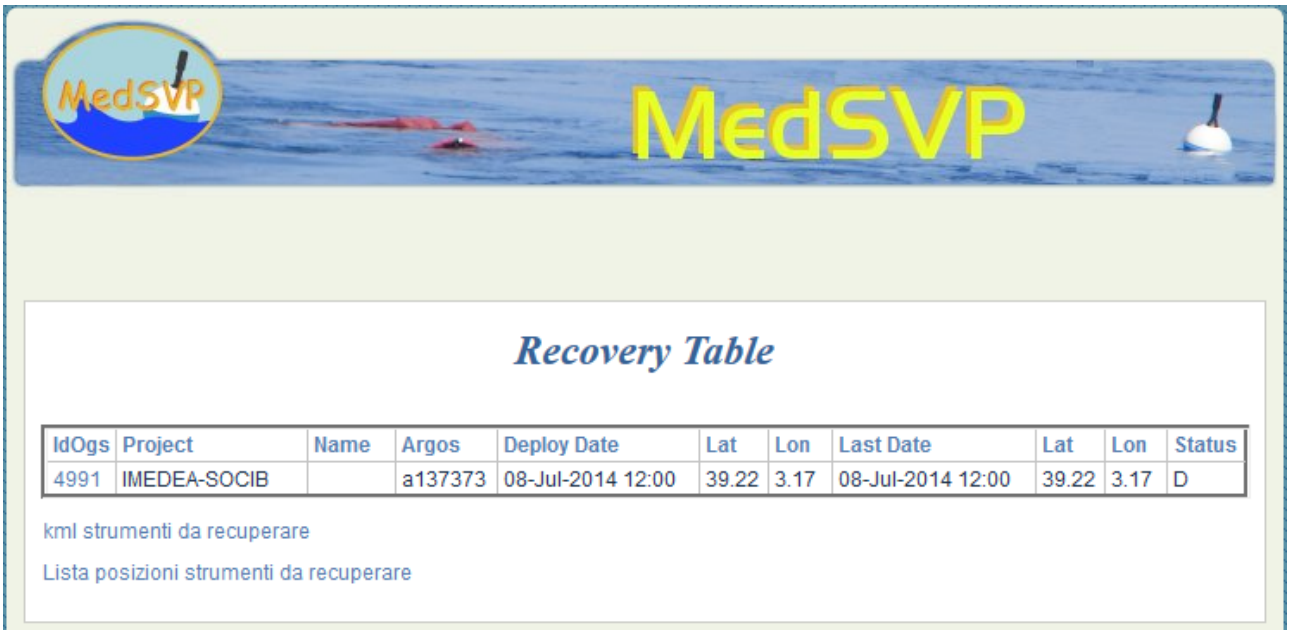
Il codice crea al volo lo script matlab che verrà eseguito subito dopo.

Per la generazione delle figure png o kml, di gruppo o si singoli strumenti:

```
## --- da 8192 a 131072 --- ##
$COMMONDIR/list_instr_mfile_multi_rec.pl Drifter /tmp/drifter_rec.m
$PAR $MATLABR "try; run '/tmp/drifter_rec.m'; catch; end; quit" >> $STDOUT
```

Pagine web

http://nettuno.ogs.trieste.it/sire/drifter/drifter_rec.php



http://nettuno.ogs.trieste.it/sire/drifter/drifter_rec_txt.php (relativo ad un altro drifter in questo esempio)

Recover Drifter - last position

```
Strumento: b131969 Elaborato il: 05-Aug-2014 15:54:41 - Tipo: SVP - SN: 00000 - Status: A
nmsg --- Time ---------- lat --- lon --- volt h2o rssi mem ------ lat -------- lon
0665=2014-08-05 03:00:03 38.1308 1.0169 gradi: 38° 7.84800' - 1° 1.01400'
0664=2014-08-05 02:00:03 38.1287 1.0173 gradi: 38° 7.72200' - 1° 1.03800'
0663=2014-08-05 01:00:06 38.1274 1.0174 gradi: 38° 7.64400' - 1° 1.04400'
0662=2014-08-05 00:00:02 38.1269 1.0179 gradi: 38° 7.61400' - 1° 1.07400'
0661=2014-08-04 23:00:04 38.1266 1.0193 gradi: 38° 7.59600' - 1° 1.15800'
```

Creazione funzione matlab estrazioni dati drifter – da implementare

Per poter estrarre tutti i dati dal db (a comando), bisognerebbe implementare una funzione matlab avente come ingresso la query sql e in uscita una struttura contenente i vari campi ed i relativi dati.

Per i float la funzione esiste già e si chiama read_float_db

Si è iniziato a sviluppare la funzione per i drifter (read_drifter_db.m; contenuta in /storage/sire/work/drifter/matlab/) che però deve essere ancora implementata e testata.

Appendice: Postgres e shell linux

Per poter interrogare il database dalla shell è possibile usare il comando psql.
Si consiglia di leggere la relativa pagina del manuale:
http://www.skrenta.com/rt/man/psql.1.html

Ecco alcuni esempi con la relativa risposta da parte del db interrogato (il db in questione ha nome float):

```
psql -U postgres -d float -c "SELECT id_drifter, argo_mission, id_argo FROM
tbl_drifter WHERE id_argo = '131968'"
 id_drifter | argo_mission | id_argo
------------+--------------+---------
       4986 | a            | 131968
(1 row)
```

se è necessario usare l'output del comando in qualche altro script, si consiglia l'utilizzo del parametro -t che disabilita l'intestazione dei nomi dei campi.

```
psql -t -U postgres -d float -c "SELECT id_drifter, argo_mission, id_argo FROM
tbl_drifter WHERE id_argo = '131968'"
       4986 | a            | 131968
```

È possibile usare un utente “normale” (in questo caso l'utente che esegue il psql è drifter), per accedere a determinate tabelle del db:

```
psql -t -d float -c "SELECT id_drifter, argo_mission, id_argo FROM tbl_drifter
WHERE id_argo = '131968'"
```

Il divisore tra i campi e' di default “|” e per modificarlo ad esempio con un tab (è necessario rimuovere l'incolonnamento con l'opzione -A; bug psql relativo alla versione attualmente installata), va usato:

```
psql -t -d float -A -F $'\t'  -c "SELECT id_drifter, argo_mission, id_argo FROM
tbl_drifter WHERE id_argo = '131968'"
4986    a       131968
```

Per poter accedere al dato di recupero per i drifter in questione è necessario estrapolare il dato numerico in bit (in questo esempio è stato settato ad 1 il campo dati_recupero_rt per un singolo drifter).

```
psql -U postgres -d float -c "SELECT DISTINCT substring(cast(dati_recupero_rt as
int4)::bit(32) from 1 for 32) as dec FROM tbl_drifter WHERE dati_recupero_rt<>0"
               dec
----------------------------------
 00000000000000000000000000000001
(1 row)
```

```
psql -t -d float -c $'SELECT DISTINCT substring(cast(dati_recupero_rt as
int4)::bit(32) from 1 for 32) as dec FROM tbl_drifter WHERE
dati_recupero_rt<>0;'
 00000000000000000000000000000001
```

Quindi se noi volessimo ottenere solo il Xesimo valore è necessario partire dal Xesimo carattere (attenzione: Xesimo partendo da sinistra, ma 32-X+1 partendo da destra) ed estrapolarne solo uno.

Ad esempio, volendo analizzare il 14esimo bit ed ottenere il suo valore con una query, per un drifter con il valore di dati_recupero_rt settato a 262144 (2^18):

```
psql -t -d float -c $'SELECT DISTINCT substring(cast(dati_recupero_rt as
int4)::bit(32) from 1 for 32) as dec FROM tbl_drifter WHERE
dati_recupero_rt<>0;'
 00000000000001000000000000000000
```

si usa:

```
psql -t -d float -c $'SELECT DISTINCT substring(cast(dati_recupero_rt as
int4)::bit(32) from 14 for 1) as dec FROM tbl_drifter WHERE
dati_recupero_rt<>0;'
 1
```

Riferimenti/Bibliografia/Relazioni

Psql: http://www.postgresql.org/docs/8.1/static/app-psql.html