Replicatore AdEr

Progetto in CVS: /var/opt/cvs/progetti/3di.lua.dcw2mysql su cvs.3di.it

Struttura del progetto

Ci sono 2 cartelle:

- build_sync: Questa contiene uno script ANT che può essere utile in laboratorio per copiare i file LUA nella loro cartella di destinazione;
- mysql: Tutte le variazioni al DB realizzato per AdER ed un breve file di istruzioni. Ci sono dentro tutte le changes datate da applicare in ordine;
- script: Contiene il file di replica principale (dcw2mysql.lua) ils uo file di cofirurazione (con esempi ed un copia non aggiornata di quello presso AdER).

Funzionamento della replica, live & past

Avvio

Il processo viene avviato tramite Job di eXtraWay EE (Vds. /opt/3di.it/extraway/xwee/conf/jobs.conf.xml)

Il Job viene eseguito 2 volte al giorno:

- La sera verso le 19, per consumare quanto più possibile del lavoro diurno;
- La mattina verso le 7 (forse prima), per consumare quanto fatto dagli automatismi almeno fino alla mezzanotte.

Il file jobs ha un aspetto di questo tipo:

```
<job arc="acl" when="at" val="1900:*" label="every day at 19:00">
   <cmd stored="$.dcw2mysql.liveReplicator">
   </cmd>
</job>
<job arc="xdocwaydoc" when="at" val="1910:*" label="every day at 19:10">
   <cmd stored="$.dcw2mysql.liveReplicator">
   </cmd>
</job>
```

Si indica il nome (logico) dell'archivio, la modalità ("at" che indica l'avvio ad un'ora precisa), un valore temporale (l'ora, seguita da '*' per indicare tutti i giorni), un label facoltativa (la si trova nei log se serve cercala...) ed il comando, così come verrebbe eseguito da console con un XML Command.

Comandi principali

- liveReplicator: effettua la replica quotidiana a partire dall'ultima replica effettuata;
- pastReplicator: effettua replica "all'indietro" per recuperare dati del pregresso.

liveReplicator

Anche se il file dcw2mysql.lua è comune (/opt/3di.it/extraway/xwee/script) ciascun archivio ha il proprio registro dell'ultima operazione effettuata in /opt/3di.it/extraway/xwee/db/<archivio>/<archivio>.script/replica_live.xml

Il file ha questa forma:

<?xml version="1.0" encoding="UTF-8"?> <replica db="xdocwaydoc" host="127.0.0.1" port="3306" usr="root" pwd="root" date="20190705" time="164600"/>

Attributi

- db: nome dell'archivio MySQL
- host & port: dove si trova MySQL server
- usr & pwd:...
- date & time: data ed ora del momento in cui è stata fatta l'ultima replica.

La replica live seleziona tutti i record inseriti o modificati con data ed ora maggiore o uguale a quella salvata nel record. Solo se la replica si completa correttamente questi due valori vengono rettificati.

In origine il processo si bloccava ad ogni errore major, ora ci hanno chiesto di segnalarli e tirar di lungo, in questo modo il più delle volte anche in presenza di errori il processo giunge a termine e rettifica la data di ultima replica.

Il processo prevede per ciascun archivio:

- Invio di mail di inizio lavorazione;
- Invio di una mail per ogni record non replicabile per problemi "major";
- Invio di mail di fine lavorazione, con indicazione di quanto fatto e se la replica possa dirsi "conclusa" o meno.

Server coinvolti

Vecchio server di produzione SDR

- `acl`
- `xdocwaydoc`
- `xdocwaydoc-spa`

Server di produzione AdER

- `acl`
- `xdocwaydoc`

pastReplicator

Quando chiedono interventi con efficacia retroattiva, dobbiamo forzare la replica di tutto il pregresso, dell'uno o dell'altro archivio.

Anche in questo caso si usa un Job che lavora solitamente negli orari di minore attività.

La filosofia è simile a quella del `liveReplicator`, ma si basa sul file `replica_past.xml` ed ha un comando Job lievemente diverso.

```
<job arc="aequisdr" when="at" val="1930:*" label="every day at 19:30">
   <cmd stored="$.dcw2mysql.pastReplicator">
      <endTime>2350</endTime>
   </cmd>
</job>
```

Evoca il comando `pastReplicator` che viene avviato all'ora richiesta e si interrompe all'ora indicata col parametro 'endTime'.

`N.B.`: Nel meccanismo dei Job il tempo è sempre riferito al giorno corrente. Se si vuol far compiere un'attività tra le 20:00 e le 05:00 del mattino, essa va spezzata in due Job, uno dalle 20:00 ad esempio alle 23:55 ed uno dalle 00:01 alle 05:00.

Il file sul quale si basa la `replica past` è fatto così:

<?xml version="1.0" encoding="UTF-8"?> <replica db="xdocwaydoc" host="127.0.0.1" port="3306" usr="root" pwd="root" last="392778" most="2781778" first="1" step="-1000"/>

Esso si compone di:

- `db, host, port, usr, pwd`: Esattamente come il replica_live.xml;
- `last`: Documento dell'archivio dal quale far partire la replica;
- `first`: Documento dell'archivio raggiunto il quale la replica deve interrompersi;
- `step`: Quanti documenti vengono fatti di volta in volta prima di rettificare questo file;
- `most`: Non serve, me lo segno io per ricordarmi da dove sono partito.

L'esempio prevede di partire dal fondo e procedere a ritroso verso l'inizio. Questo perché solitamente i record più recenti sono quelli che gli interessa vengano replicati prima.

Di volta in volta, mentre `first` rimane invariato a `1`, `last` cambia valore di una misura pari a `step`. Mettendo "step" con valore positivo, la replica non procede da "last" verso "first" ma in senso inverso ed inversamente verrà rettificato "first" e lasciato invariato "last".

Come il processo di replica live, la `replica past` manda una mail di inizio lavori ed una di fine lavori.

Quando `first` e `last` si equivalgono, la replica si interrompe ed invia una notifica via mail per chiedere che il Job venga inibito.

dcw2mysql.lua

Questo script viene lanciato su due tipologie di archivio, `ACL` e `DCW`.

Che si usi `liveReplicator` o `pastReplicator`, il processo compie replica di tutti i record previsti in un elenco (da selezione o da numeri fisici), manda mail di inizio e fine lavori notificando se ci sono stati errori bloccanti o meno, errori quindi che chiedano un intervento manuale.

Vediamo ora come funziona il processo di replica vero e proprio.

La modalità di `replica op` indica in che modo si doveva compiere la replica[1].

- In assenza di connessione attiva ne apre una nei confronti di MySQL sulla base dei dati raccolti;
- Imposta l'encoding UTF ed apre la transazione;
- Controlla di che tipo di record si tratta ed evoca uno specifico replicatore per ciascuno di essi, distinguendo le modalità "ins" e "mod", trattate nello stesso modo, dalla modalità "del". In realtà solo la modalità "mod" viene usata;
- Ciascuna funzione di replica acquisisce elementi ed attributi dal record e popola una tabella principale, poi, in seguito, procede con gli elementi ripetibili, in modo analogo, e popola le tabelle correlate;
- Tutte le operazioni di inserimento sono di fatto delle UPSERT che vengono composte in questo modo:

1. Man mano che si aggiungono campi (elementi o attributi) si popola una tabella con nome colonna e valore ed

eventualmente una tabella parallela con i valori “if null”;
2. Il valore della colonna può essere popolato in modo standard (`addColunm`), con un valore calcolato (`addValueColumn`) o in modo tradotto (`addTranslatedColumn`).

- Ciascuna delle `tre modalità` prevede:

1. Il record XML;
2. Eventuale nodo dal quale prendere il valore con percorso relativo;
3. xPath da cui prendere il valore (`addColumn`) o il valore già calcolato (`addValueColumn`);
4. Nome della colonna che lo ospiterà;
5. Modalità di conversione (standard, bool, num, date);
6. La tabella dove introdurre l’espressione, la tabella dei valori nulli, ed il default;
7. Solo per il tradotto, una tabellina dei valori tradotti ed un default.

- Una volta preparata la tabella coi valori (`doneTab`) e quella dei nulli (`nilTab`), con alcuni comandi (`table.concat`) si compone il comando vero e proprio. Applicando un trucco “ ON DUPLICATE KEY UPDATE#”, si concatena una seconda volta lo stesso set di campi ed i relativi valori da annullare.

Segue altro trucco per sostituire il termine UPDATE# che serve a togliere la prima virgola e poi si annullano i valori da non toccare. * Viene eseguita l’operazione su MySQL (`con:execute()`), collezionato il risultato e si va avanti. * A seguire, per tutte le aree ripetibili, si segue la stessa filosofia, usando dove necessario il nodo ripetuto dal quale prendere un valore e così via. Praticametne in tutti i casi si procede alla cancellazione delle righe della tabella che si andrà a popolare per tener conto dei casi di rimozione. * Al termine del processo di replica, la funzione torna:

1. `done`: true o false per dire se è andato ao buon fine o c’è stato un errore;
2. `outMsg`: eventuali informazioni supplementari da notificare;
3. eventuale parametro `bool` per dire se la condizione d’errore è “minor”, qualora ‘true’. In tutti gli altri casi se done=false il livello d’errore sarà major.

Per fare le attività dette si avvale ampiamente di `mysqlutils.lua` che appartiene al set standard di `script lua di eXtraWay (SE/EE)`.

Trattamento degli errori

Vediamo il file `dcw2mysqlcfg.lua`

```
-- DocWay to MySQL sample configuration

local dcw2mysqlcfg = {
  onErrorAction = {major='mail', minor='skip', doubleAttachment='skip', missingAttachment='skip',
doubleFolder='skip'}, -- Value can be 'stop', 'mail', skip'
  notifyTo =
{to='vincenzo.lauria@agenziariscossione.gov.it;paolo.ciappi@agenziariscossione.gov.it'},
  mailTemplate = 'Something wrong replaying on #db# using #usr#@#host#:#port#',
  doubleAttachmentMailTemplate = 'A record contains a double rif to the same attachment.\nYou
don\'t need to do anything but please pay attention to attachment #attach# on record #nrecord#',
  doubleFolderMailTemplate = 'A record contains a double rif to the same folder.\nYou don\'t need
to do anything but please pay attention to folder #cod_fasc# on record #nrecord#',
  missingAttachmentMailTemplate = 'A record contains a rif to an empty attachment.\nYou don\'t
need to do anything but please pay attention to record #nrecord#',
  missingCfgReplicaFile = 'Can\'t find file #stepFile#.\nReplica can\'t be performed.',
  pastReplicaCompleted = 'Preceding replica as from file #stepFile# complete.\nPlease stop job.',
  mailOpt = {
    from = 'dcw2mysql@agenziariscossione.gov.it', --, -- Admin email it will figure in mail as
sender mail
    from_label = 'DocWay 2 MySQL Replicator',
    -- user = nil, -- Admin smtp user name if login enabled nil otherwise
    -- password = nil, --Admin smtp password if login enabled nil otherwise
    -- server = 'wrong.smtp.com', -- smtp server host
    -- port = 25, -- smtp server port {25 | 465}
    -- ssl = true --true if there is an ssl communication { true | false }
   },
   codammaoo = 'ADERISC'
}

return dcw2mysqlcfg
```

In particolare ci interessa la tabella `onErrorAction` che ci dice come comportarci nei diversi casi trattati.

- `major`: errori di primaria importanza, sono errori per i quali il processo dovrebbe interrompersi;
- `minor`: errori secondari, possono non interrompere il processo ma è utile che non vengano ignorati;
- `doubleAttachment`: Può essere ignorato (skip) o manda mail di notifica che il record presenta più volte lo stesso file allegato. Se non è né skip né mail da errore;
- `missingAttachment`: Quando pur in presenza di un elemento di tipo "xw:file" manca il nome dell'allegato (attributo 'name'). Stesso comportamento di cui sopra;
- `doubleFolter`: segnala che il record è stato collegato a più fascicoli. Si attiva solo in modalità mail.

Per ciascuno di essi i `valori ammessi` sono:

- `stop`: interrompe il processo, causa una segnalazione d'errore nella mail di fine lavorazione;
- `mail`: invia una mail di notifica agli utenti configurati alla voce "notifyTo";
- `skip`: ignora l'errore e si limita a segnalarlo nel log.

In tutti i casi d'errore che non vengono trattati come `skip` o `mail` (condizione che comporta che l'errore viene eventualmente notificato ma non interrompe il flusso del processo), lo stesso errore viene qualificato come major o minor così che il replicatore possa agire di conseguenza, seguendo quanto impostato nel file di configurazione.

Se c'è errore e per quella tipologia d'errore (major/minor) si usa la modalità `stop` il processo si interrompe e notifica quanti record ci sono ancora da elaborare rispetto alla selezione/elenco iniziale.

Attualmente hanno richiesto di non essere interrotti neanche in caso di errore major perché preferiscono avere alcuni record in meno che si possono anche replicare a mano, che dover rinunciare a molti record replicabili.

customReplicator

Compie esattamente lo stesso lavoro del `liveReplicator` ma non calcola la query sulla base del file `replica_live.xml` ne lo aggiorna. Serve a replicare manualmente dei record sui quali si sia interventi per qualche ragione dopo, ad esempio, che la replica live ha segnalato errori sui record, per non dover attendere la replica automatica serale o mattutina, ma ovviamente richiede una query che li identifichi.

Il comportamento è del tutto analogo, compreso l'invio delle mail di inizio e fine lavorazione.

```
<cmd stored="$.dcw2mysql.customReplicator">
        <query>[/doc/@nrecord]=...</query>
</cmd>
```

pecReplicator

Processo "super-custom" fatto per replicare solo i record di struttura esterna o persona esterna per le quali fosse necessario aggiornare le pec in ACL.

Si poteva fare con `pastReplicator` ma ci è stato chiesto di farlo in tempi brevi. Rettificare solo i record in cui c'è questo tipo d'informazione era più veloce che replicare l'intero archivio che può richiedere anche qualche giorno o per lo meno molte ore.

Questo processo può essere ignorato o preso a spunto per altre customizzazioni.

Nuovi campi

L'introduzione dei nuovi campi si fa così:

- Accedere all'archivio con MySQL WorkBench.
- Applicare le varianti (es. aggiunta campo "pippo"). Impostare 'pk' per primary key, 'nn' per not null, 'uq' per unique index, 'b' per binary e così via...
- e quando si è pronti ad eseguirle, ('Apply') copiare il comando che viene presentato ed eseguirlo.
- Creare un nuovo file changes_<data>.sql nella cartella 'mysql' del progetto ed applicare al suo interno il comando copiato.

N.B.: dal momento che la versione di MySQL che siamo soliti usare noi (5.7.X) differisce da quella che usano loro (5.6.35), i comandi che vengono segnalati come 'CHANGE COLUMN' devono essere invece espressi come 'MODIFY'. (Vds. changes_20191028.sql).

- Raccolte le istruzioni, variazioni al file dcw2mysql.lua (ed eventuali a dcw2ysqlcfg.lua), il file .sql e quanto altro dovesse servire, si invia tutto ad AdER per l'applicazione. Si suggerisce di applicare il nuovo file dcw2mysql.lua/dcw2mysqlcfg.lua solo dopo aver compiuto gli inteventi sul DB MySQL.

L'accesso al DB MySQL del cliente

Il DB MySQL del cliente si trova su un set di macchine verso le quali non abbiamo un accesso diretto. Abbiamo però la possibilità di accedervi passando dal server di produzione ed aprendo opportuni tunnel.

Vale quindi la pena di aprire un tunnel con la porta desiderata localmente[2] che, una volta collegati al `Server di Produzione AdER` punti al loro `server MySQL (10.55.240.194:3306)`.

Il collegamento con strumenti quali MySQL Workbench darà accesso e visibilità direttamente dell'archivio del cliente. Le credenziali d'accesso sono le stesse registrate nei file `replica_live.xml/replica_past.xml` del cliente.

Gestione degli scarti

A intervalli regolari ADER apre delle segnalazioni relative agli scarti del replicatore.

I possibili `problemi` possono essere i seguenti:

- `Caratteri non standard` nell'oggetto del documento (emoticons varie oppure lettere accentate o altro); per correggere occorre rimuovere il carattere.
- `Mancanza` di alcuni `dati obbligatori`, soprattutto relativamente ai campi custom relativi ai repertori Contenzioso Atti Introduttivi e Atti Successivi; ad esempio spesso capita che sia presente solo il campo data_notifica_ricorso e in questo caso la replica del record va in errore in quanto mancano tutti gli altri dati obbligatori per valorizzare la tabella mysql
- Relativamente al repertorio Atti Successivi è capitato che il campo `numero_pratica_al` contenesse più di un valore. La cosa è stata gestita lato script lua, in quanto il valore del campo viene splittato a patto che il separatore dei campi sia lo spazio; se mettono una virgola come separatore non funziona

[1] Lo script era stato originariamente concepito per essere usato col replicato di eXtraWay EE. Col replicatore sono previste altre modalità di replica, quali "ins", "del", e "attach". In questo caso si opera sempre in modalità "mod".

[2] MySQL utilizza la porta 3306, ma per essere sicuri di non interferire con il proprio MySQL locale e soprattutto di non trovarsi ad osservare un archivio pensando che sia quello del cliente (o quello locale e vice versa) e farci sopra interventi non appropriati, conviene scegliere per il tunnel una porta diversa, ad esempio 3308