

DATASERVICE

INDICE

- [Filosofia del dataservice](#)
 - [Confronto con ODBC "tradizionale"](#)
 - [Struttura](#)
 - [Scelte orientate alla "fruibilità"](#)
 - [Altre scelte di fondo](#)
 - [Rischi potenziali](#)
- [Installazione database](#)
- [Amministrazione del database con pgAdmin III](#)
- [Creazione utente del database "primula"](#)
- [Creazione database "primula"](#)
- [Restore database "vuoto"](#)
- [Inizializzazione tabella AZIENDE](#)
- [Driver ODBC PostgreSQL](#)
 - [Accesso via rete](#)
 - [Creazione utenti database "read only"](#)
- [Configurazione lato Primula](#)
- [Server di pubblicazione](#)
 - [Server di pubblicazione "silenzioso"](#)
- [Test funzionamento](#)
- [Popolare le tabelle](#)
- [Controllo allineamento dati](#)
- [Casistiche speciali e particolari](#)
- [Backup \(dump/vacuum\) e restore](#)
- [Logiche di programmazione](#)

E' stata implementata una nuova procedura finalizzata alla pubblicazione della base dati di Primula all'interno di un database relazionale di natura gestionale, denominata **dataservice (DS)**.

Filosofia del dataservice [\[Indice\]](#)

La filosofia principale che spinge ad implementare il dataservice è la volontà di rendere disponibili e fruibili i dati di Primula per elaborazioni terze al gestionale stesso.

Gli ambiti principali di utilizzo di questa struttura dati è quello del passaggio dati ad altre applicazioni gestionali, della reportistica, della elaborazione statistica e dell'analisi dei dati con tecniche di Business Intelligence (BI) – o similari.

Confronto con ODBC "tradizionale"

Ci sono una serie di considerazioni importanti che inducono ad escludere che la struttura del DS costituisca il naturale sostituto dell'ODBC tradizionale, sebbene risulti evidente che i vincoli ed i limiti del "vecchio" strumento – culminati con

l'impossibilità di installazione dello stesso su sistemi operativi di tipo server e/o a 64 bit – hanno trovato naturale risposta nel nuovo strumento.

Sta di fatto che DS **non** ha le seguenti caratteristiche altresì presenti nell'ODBC "tradizionale":

- **Disponibilità di tutte le tabelle** e di tutti i campi di ciascuna di esse: in DS sono presenti solo le tabelle esplicitamente implementate
- **Disponibilità dei dati tecnicamente in tempo reale:** DS raccoglie i dati con una tecnica che passa da un server di pubblicazione; gli utenti di Primula accodano eventi del tipo "dato inserito", "dato modificato", "dato cancellato" per le varie tabelle, ed il server di pubblicazione esegue di conseguenza questi "comandi" evocati dalle attività dei singoli utenti, trasformandoli in variazioni nelle tabelle del database. Non si tratta quindi di un aggiornamento che possa definirsi informaticamente in tempo reale, sebbene si avvicini molto a questo concetto
- Sebbene il dizionario dati dell'ODBC venisse rilasciato in modalità "read only", come tipo di strumento consente anche l'accesso in scrittura dei dati, cosa non possibile con DS

Per contro, esistono parecchie **motivazioni "a favore" del nuovo servizio**, tali da renderlo una evoluzione fondamentale nei processi di scambio dati.

Con la rel 7.1 di Primula, non viene più rilasciato il dizionario dati dell'ODBC tradizionale aggiornato.

Struttura

In forza della logica gestionale del database, la struttura è riconducibile in modo abbastanza fedele alle corrispondenti tabelle presenti in Primula; i dati sono esposti in linea di massima in terza forma normale, senza ridondanze; o perlomeno rispecchiano le medesime scelte strutturali delle tabelle native di Primula.

Non sono stati attivati vincoli di integrità referenziale dei dati, potenzialmente gestibili dal motore del database.

Scelte orientate alla "fruibilità"

In molti dettagli – più o meno rilevanti – si denota quanto abbia pesato nella configurazione del database il pensarlo **fortemente orientato alla fruibilità** da parte di terze parti.

In particolare:

- Ogni tabella riporta un campo univoco nel database, su ciascun record, denominato **UUID** e l'informazione del nome **utente** di Primula e **data/ora** dell'ultima variazione. Queste informazioni sono gestite direttamente dal motore del database, tramite dei trigger. Di fatto **è sempre possibile sapere quali record sono stati modificati o inseriti dopo un dato momento**
- E' presente una tabella **LOG** che riporta l'insieme degli **UUID** di ciascuna tabella e le relative sequenze di **utente/data/ora** delle variazioni.

Anche da questa tabella è possibile ricostruire tutte le variazioni intervenute sulle tabelle gestionali. A differenza della funzionalità precedente (LOG), che indica lo stato di fatto attuale del record della tabella, questa consultazione consente di ricostruire la sequenza di modifiche dei record. Chiaramente non è possibile sapere quale campo è stato modificato all'interno del record

- Alcune tabelle di tipo "**prolunga**" che in Primula sono costruite su file differenti aventi la medesima chiave, in DS **sono riportate tutte nel record della tabella principale**. Solo a titolo di esempio, i dati contenuti nel file di "prolunga" dei clienti e fornitori APPCLF, sono presenti in DS come campi della unica tabella CLIFOR
- Per alcune **tabelle** che in Primula costituiscono (per vincolo strutturale o fiscale, oppure per semplice consuetudine) oggetto di **eliminazione periodica**, in DS si è provveduto ad implementare un sistema di **storicizzazione**, tale per cui, sebbene i dati continuino ad essere eliminati lato Primula, vengono mantenuti e conservati in DS. Evidentemente occorre provvedere a strumenti come report o pannelli di visualizzazione qualora necessiti una consultazione degli stessi.

Ad esemplificazione, il caso classico è quello delle fatture di vendita.

Quando ad inizio anno vengono eliminate da Primula le fatture di pari numerazione dell'anno precedente, le stesse vengono storicizzate sul DS e rimangono quindi consultabili per le necessità future. Questa implementazione è stata fatta tipicamente inserendo fra i campi della chiave "naturale" della tabella un campo **serie**, tipicamente costituito dall'anno di emissione del documento

- Sono state realizzate due viste "**elenco_tabelle**" ed "**elenco_campi**" tramite le quali è possibile accedere all'elenco commentato delle tabelle disponibili ed ai relativi campi, **sempre commentati**
- La maggior parte degli archivi non anagrafici (di tipo "movimenti" o di tipo "testate/righi") sono stati strutturati con una chiave principale di tipo ID (numerico univoco), oltre alla chiave "naturale" corrispondente a quella già presente nel gestionale (definita come chiave esterna).
- Alcune tabelle sono state rese uniformi, aggregando dati che nel gestionale sono sparsi in strutture differenti. Il caso emblematico è quello della tabella dei movimenti contabili, che in DS sono riportati in unica tabella (sia generale che sezionale, sia esercizio corrente che vecchi esercizi).
- Alcuni dati "ragionati" che non sono fisicamente presenti fra i dati di Primula, sono presenti e mantenuti aggiornati all'interno di DS, in modo che possano essere utilizzati senza applicare alcun algoritmo da parte del fruitore delle informazioni. Un esempio è costituito dai dati del "consegnato" presente nei righi di dettaglio degli ordini a programma clienti e fornitori.

Altre scelte di fondo

Il database relazionale che è stato scelto di implementare è **PostgreSQL**. La scelta è ricaduta su questo strumento a partire dalla esperienza maturata in Agomir sui progetti Integra.

Inoltre, in aggiunta al fatto che lo strumento è gratuito, si è dimostrato di robustezza, affidabilità e efficienza di prim'ordine.

Essendo un database che "vive" in simbiosi con i dati reali di Primula, si propone come configurazione consueta di installare il database sul medesimo server di Primula.

Questo anche nel caso esista già un altro server nella rete su cui è installato Postgres, quale il server Integra – ad esempio. Questo ci consente anche di slegarci da vincoli reciproci di release di database.

Anche nel caso in cui in Primula esistano più aziende gestite nella medesima area FL, si consiglia in DS di installare istanze divise e distinte di database, fatto salvo il caso di esigenze di analisi sul "consolidato".

In ogni caso tutti i record di tutte le tabelle sono identificati da un codice azienda alfanumerico univoco.

Rischi potenziali

Trattandosi di un database fisicamente separato dai dati "veri" di Primula, non è impossibile ipotizzare che si possano creare dei disallineamenti. Questi potrebbero ricadere in almeno due casistiche:

- È intervenuto un evento che ha procurato il mancato aggiornamento della informazione su DS da parte del server di pubblicazione
- Uno o più programmi che aggiornano una specifica informazione non accodano l'evento nella coda del server di pubblicazione
Questa eventualità diviene più plausibile nel caso di presenza di personalizzazioni alla procedura standard, sviluppate per un progetto o un cliente specifico

E' per questo motivo che vengono rese disponibili delle funzioni di conciliazione dei dati.

Installazione database [[Indice](#)]

Procedere con l'installazione di PostgreSQL a partire dal programma di installazione rilasciato nel CD di Primula o scaricato dal sito <http://www.postgresql.org/download>

Su macchine a 64 bit non ci sono controindicazioni ad installare la versione del programma corrispondente.

DS è funzionante su una versione di database >= a 9.0

In fase di installazione viene richiesta la cartella di installazione del **programma** (solitamente può essere confermata quella proposta), e la cartella di installazione dei **dati**; in caso di disponibilità di dischi separati e specifici per programmi e dati, la

cartella dati è bene che venga impostata sul disco preposto, per fare in modo che poi possa essere oggetto della procedure di backup.

Viene anche richiesta la password per l'utente di sistema "postgres". Annotarsi la password che viene impostata. "postgres" viene creato anche come utente di Windows.

Possibili problemi:

In fase di installazione, su alcuni sistemi – specialmente su sistemi a 64 bit o in presenza di particolari restrizioni per le policy degli utenti – viene segnalato un errore sui permessi per la creazione della cartella dei dati. In questi casi la procedura non va a buon fine.

"problem running post-install step. installation may not completed correctly"

In questo caso, eseguire le seguenti attività:

- cancellare l'utente di Windows "postgres" e ricrearlo (impostando la password per l'utente nel secondo comando)

```
net user postgres /delete  
net user postgres [password] /add
```

- attribuire gruppo Administrator e Power Users all'utente postgres

```
net localgroup "Administrators" "postgres" /add  
net localgroup "Power Users" "postgres" /add
```

- su cartella dei dati (es: c:\postgresql) forzare **controllo completo** per l'utente postgres (da tasto dx + proprietà)

- da "esegui" aprire prompt dei comandi con i diritti dell'utente postgres

```
runas /user:postgres cmd.exe
```

- lanciare programma di installazione dal prompt dei programmi, digitando il percorso completo del file di installazione (es: dal CD di Primula)

- terminata la procedura di installazione, togliere utente postgres da gruppo Administrator

```
net localgroup "Administrators" "postgres" /delete
```

Nella fase finale della procedura di installazione viene data la possibilità di accedere alle "Application stack builder"

Da questa sezione è possibile selezionare l'installazione del driver ODBC.

Amministrazione del database con pgAdmin III [\[Indice\]](#)

pgAdmin III è un programma con interfaccia grafica dedicato alla amministrazione del database.

Da questa procedura è possibile eseguire molte operazioni fondamentali di amministrazione.

La maggior parte delle attività che possono essere fatte da questa procedura e dalle sue maschere grafiche, è altresì eseguibile tramite il **pannello SQL** da cui è possibile inserire ed eseguire molteplici attività.

Installando le versioni più recenti del database, può accadere che pgAdmin III non abbia ancora disponibile la traduzione in italiano.

E' indispensabile da parte dell'amministratore di DS acquisire una buona dimestichezza con l'utilizzo dello strumento: tale abilità potrà servire in molti casi.

In questo documento non viene fornita nessuna indicazione sulle logiche di funzionamento di questo programma; altresì questo programma andrà utilizzato per eseguire i comandi SQL riportati nelle varie funzioni.

Le istruzioni fondamentali sono:

- lanciare pgAdmin III
- selezionare (doppio click) l'istanza di database, selezionando utente e password (es: PostgreSQL 9.3 (localhost:5432))
- Aprire la cartella Database e selezionare il database su cui operare
Dall'interno di questa cartella è possibile navigare ad albero accedendo alle singole tabelle/campi del database
- Aprire il pannello SQL per eseguire comandi sql
- Quando si eseguono comandi che alterano il database, eseguire la refresh per visualizzare i dati di struttura aggiornati

Creazione utente del database "primula" [\[Indice\]](#)

```
CREATE ROLE primula LOGIN  
PASSWORD '[password] '  
SUPERUSER INHERIT CREATEDB CREATEROLE REPLICATION;
```

Impostare e memorizzare la [password] dell'utente.

Il comando sql è disponibile nel file **sam\dsv\sys\user_primula.sql**

L'utente "primula" è l'utente tramite il quale verranno eseguite tutte le operazioni di I/O sul database da parte del server di pubblicazione (tramite ODBC).

ATTENZIONE: per poter eseguire via SQL questo comando, occorre (come in tutti gli altri casi) selezionare preventivamente un database; non essendo in questo punto della procedura ancora stato creato il database specifico "primula", cliccare sul database di default "postgres". I ruoli degli utenti non sono specifici di un database, perlomeno di default, ma possono essere utilizzati per qualsiasi database.

Creazione database "primula" [\[Indice\]](#)

```
CREATE DATABASE primula  
WITH OWNER = primula  
ENCODING = 'UTF8'  
TABLESPACE = pg_default
```

```
LC_COLLATE = 'Italian_Italy.1252'  
LC_CTYPE = 'Italian_Italy.1252'  
CONNECTION LIMIT = -1;
```

Restore database "vuoto" [\[Indice\]](#)

- Da pgAdmin III evidenziare il nuovo database "primula" (creato con la funzione precedente)
- Con "tasto dx" selezionare la funzione "restore"
- Sfogliare la cartella e selezionare il file **sam\dsv\db\71_custom.backup**
- Impostare il ruolo dell'utente "primula"
- Confermare l'operazione di restore
- Controllare nel pannello dei messaggi che non vengano visualizzati messaggi di errore

Inizializzazione tabella AZIENDE [\[Indice\]](#)

```
INSERT INTO aziende (azi_codice, azi_descr, azi_pri_azi, azi_pri_fl,  
azi_opc_dtape, azi_opf_dtape, azi_mmg_dtape)  
VALUES ('AZI', 'descrizione', '001', 'F:\dati\DEMO\71\fl', '2014-01-  
01', '2014-01-01', '2014-01-01')
```

La tabella aziende è fondamentale per abilitare un'area di lavoro / azienda di Primula nella gestione del DS

I campi fondamentali sono:

- azi_codice (nell'istruzione SQL precedente valorizzato con 'AZI'): rappresenta il codice azienda alfanumerico (3 caratteri) all'interno di DS. Viene utilizzato come identificativo di ogni record inserito in DS in tutte le tabelle
- azi_pri_azi (nell'istruzione SQL precedente valorizzato con '001'): questo valore deve essere il medesimo che verrà inserito nella tabella di configurazione DSVMAP

Driver ODBC PostgreSQL [\[Indice\]](#)

Il driver ODBC di Postgres è necessario:

- sulla postazione su cui verrà attivato il server di pubblicazione SRVDSV, che avrà il compito di aggiornare il database (consigliato: sul server di Primula)
- su qualsiasi posto di lavoro che intenda usufruire dei dati del DS per elaborazioni di qualsiasi sorta (tipicamente in sola lettura)

Può essere installato o dalla sezione "Application stack builder" contenuto nel programma di installazione del database, oppure dall'apposito programma di installazione specifico, contenuto nel CD di installazione di Primula o scaricabile dal sito <http://www.postgresql.org/ftp/odbc/versions/msi>

In seguito occorre **configurare il DSN**

Per effettuare questa attività occorre entrare negli **strumenti di amministrazione** di Windows e quindi nelle **origini dati ODBC**.

Nota particolare:

Primula esegue gli accessi al database via ODBC nel contesto a 32 bit.

Per le macchine a 64 bit, tale contesto a 32 bit deve essere configurato – relativamente alla definizione del DSN – non dagli strumenti di amministrazione (da cui vengono resi visibili i DSN del contesto a 64 bit), ma lanciando a mano il programma **C:\Windows\SysWOW64\odbcad32.exe**

Il DSN deve essere definito utilizzando il driver di **Postgres UNICODE**, all'interno della cartella dei **DSN di sistema**.

I dati da inserire sono:

- datasource: primula
- database: primula
- server: localhost (oppure IP del server su cui è installato il database)
- username: primula (oppure utente specifico creato per gli accessi read_only)
- password: [password]

Tramite il pulsante "test" è possibile verificare che la connessione con il database sia stabilita correttamente.

Accesso via rete:

Quando l'accesso ad database avviene via rete, per esempio nel caso di una applicazione che utilizza i dati del dataservice per reportistica o analisi dei dati, è necessario che il processo di gestione del database sia configurato per abilitare l'elenco degli IP autorizzati all'accesso.

Questa configurazione avviene modificando il file **pg_hba.conf** che si trova nella cartella dei dati del database (es: D:\PostgreSQL\9.3\data)

#	TYPE	DATABASE	USER	ADDRESS	METHOD
host		all	all	127.0.0.1/32	md5

L'indicazione 32 nella colonna dell'indirizzo va inteso come **mascheratura binaria della classe di indirizzi** (logica permission Unix-like) del tipo 255.255.255.255; quindi è abilitato di fatto il solo indirizzo 127.0.0.1 (localhost).

In queste condizioni nessun utente di rete potrà accedere al database. Solo localhost.

È possibile duplicare la riga ed inserire un nuovo ed ulteriore indirizzo di rete specifico abilitato.

In altri casi è possibile modificare la mascheratura impostando valori diversi da 32:

```
/32 mask 255.255.255.255 (solo IP specificato)
/24 mask 255.255.255.0
/16 mask 255.255.0.0
```

```
/8 mask 255.0.0.0
/0 mask 0.0.0.0
```

Quando si modifica il file pg_hba.conf è necessario riavviare il servizio Postgres del database per rendere attiva la modifica (dai servizio di Windows oppure da pgadmin III / istanza database / strumenti / ricarica la configurazione).

Creazione utenti database "read only"

Correttezza e prudenza prevede che l'accesso ai dati del dataservice da procedure esterne al gestionale avvenga tramite utenti specifici creati ad hoc e per i quali è possibile gestire i diritti, quantomeno garantendo la modalità "read only"

Questi utenti possono essere creati da pgAdmin III con i seguenti comandi SQL (selezionare il database primula):

```
CREATE ROLE user_name LOGIN
PASSWORD '[password]'
NOSUPERUSER INHERIT NOCREATEDB NOCREATEROLE NOREPLICATION;
GRANT SELECT ON ALL TABLES IN SCHEMA public TO user_name;
```

user_name e [password] vanno definite e memorizzate.

La GRANT sulle tabelle può eventualmente essere gestita in modo più puntuale, qualora all'utente non si desidera rendere visibile tutto il dataservice

Configurazione lato Primula [\[Indice\]](#)

La configurazione lato Primula prevede:

- modifica del file di mappatura **F:\programmi\71\map\srvmap** per abilitare il tipo evento DSV

```
[tipo]      DSV - 9 SRVDSV      Data Service
```

SRVDSV sta ad indicare che verrà previsto un server di pubblicazione con questo nome, specifico e dedicato (scelta consigliata).

- Copia del file **F:\programmi\71\sam\dsv\map\dsvmap** in **F:\programmi\71\map**

Modificare tale file per indicare:

```
###      azi path          _1_ 2
[azi]    001 F:\dati\DEMO\71\fl  AZI -

###      #---- dsn|user.password -----#
[odbc]   primula|primula.primula
```

La direttiva [azi] deve riportare i riferimenti all'azienda e all'area di lavoro abilitata; **nel parametro "_1_" occorre indicare il valore del codice azienda inserito nella tabella del database "aziende" nel campo "azi_pri_azi"**

La direttiva [odbc] fa riferimento al nome del DSN, al codice utente e alla password; codice utente e password possono essere omessi, nel qual caso vengono utilizzate le credenziali memorizzate nel DSN ODBC

Server di pubblicazione [\[Indice\]](#)

La configurazione del server di pubblicazione per le attività di DS segue le regole generali di configurazione dei server di pubblicazione.

Di per sé non ci sono controindicazioni ad utilizzare un unico server di pubblicazione in comune fra documenti e dataservice.

Altresì ci sono parecchie considerazioni che portano a consigliare di configurare un **server di pubblicazione specifico e dedicato**, sebbene questo comporti l'**impiego di una licenza di Primula aggiuntiva**:

- Alto numero di eventi che deve smaltire
- Possibilità di configurazione "silenziosa" (come viene specificato più avanti in questo documento), modalità che azzerà i tempi di latenza fra un evento e il successivo e quindi abbatterà significativamente i tempi complessivi di elaborazione degli eventi
- Possibilità di esecuzione a sessione disconnessa

Procedere quindi con le seguenti attività:

- Creare nella tabella server (menu SRV000 scelta 04) il codice server **SRVDSV**
Inizializzare il percorso del file della coda specifico **F:\sys\71\uti\SRVDSV**
Impostare un tempo di frequenza del polling basso (**3-5 secondi**)
Accertarsi che alla voce DSV della tabella di mappatura SRVMAP sia stato indicato esplicitamente di utilizzare il server di pubblicazione SRVDSV
- Configurare il file **F:\profile\71\primula.cfg** con la entry per il server di pubblicazione

```
# lancio server DATASERVICE
[SRVDSV]
GRGRP      P  area|azienda|ufficio|tse
LOGNAME    Admin
GREXEC     P  vsrv000 SRVDSV
```

Ricompilarlo con il comando creacfg.exe

- Creare la cartella temporanea **F:\sys\71\tmp\SRVDSV**
- Creare l'icona di lancio del server di pubblicazione
- Provare ad attivarlo per verificare il corretto funzionamento

Server di pubblicazione "silenzioso"

- Configurare il file **F:\profile\71\primula.cfg** con la entry per il server di pubblicazione

```
# lancio server DATASERVICE (silenzioso)
[SRVDSVr]
WS          SRVDSV
GRGRP      P  area|azienda|ufficio|tse
LOGNAME    Admin
GREXEC     P  /R vsrv001r SRVDSV
```

Ricompilarlo con il comando creacfg.exe

Il server di pubblicazione silenzioso, nel momento in cui viene lanciato, non fa comparire a video nessuna finestra. Questa modalità, come già documentato sopra, ha una serie di vantaggi.

Ha altresì lo svantaggio che proprio per il fatto che "non si vede", è di più difficile controllo.

L'unico modo per vedere se è in funzione è di controllare dal "monitor server di pubblicazione".

Per fermare la sua esecuzione, eseguire l'icona del server "in chiaro" e forzare lo spegnimento.

Normalmente, sul desktop è necessario e opportuno avere la sola icona di lancio "in chiaro", mentre nelle operazioni pianificate è possibile inserire la modalità di lancio "silenziosa".

- Creare la cartella **F:\sys\71\tmp\SRVDSVr**
Se non si crea questa cartella, il server di pubblicazione "silenzioso" non parte

Test funzionamento [\[Indice\]](#)

Per verificare se tutto funziona correttamente, **è opportuno eseguire le seguenti fasi di test:**

- Accedere a Primula dalla medesima postazione o sessione da cui si esegue il server di pubblicazione del DS, entrare nel menu SER000 e aprire il **prompt dei comandi**
- Eseguire il comando:

```
grrun pgmrun dsv-test
```

Se tutto risulta configurato correttamente, l'esecuzione va a buon fine senza segnalare nulla; in caso contrario, viene segnalato un errore di accesso ODBC ai dati del DS.

In caso di errore, le cose da controllare sono:

- corretto funzionamento del test di connessione dalla configurazione del DSN ODBC
- coerenza del nome del DSN e di quanto dichiarato nel file DSVMAP
- coerenza dei dati della direttiva [azi] del file DSVMAP
- esistenza del record nella tabella aziende di DS
- coerenza del codice di trascodifica azienda presente nel file DSVMAP e il valore del campo "azi_pri_azi" della tabella aziende del DS
- correttezza del nome utente/password nel file DSVMAP

- Verificare che il server di pubblicazione del DS sia fermo ed entrare in una gestione archivio, per esempio l'anagrafica clienti, e modificare un record, confermando ti dati così come sono
- Entrare nel menu SRV000 nel monitor coda documenti e verificare che **l'evento di pubblicazione risulti accodato**
- Attivare il server di pubblicazione; verificare che **l'evento venga smaltito senza errori**
- Accedere al programma di amministrazione pgAdmin III e visualizzare che il record dell'anagrafica che si è provveduto a modificare risulti correttamente inserito

Popolare le tabelle [\[Indice\]](#)

L'attività di **popolamento iniziale** delle tabelle del database DS è una attività sistemistica di amministrazione fondamentale e preliminare a qualsiasi utilizzo dei dati.

Non esiste un menu specifico da cui effettuare questa attività; occorre accedere al menu SER000 / prompt dei comandi da qualsiasi postazione Primula e lanciare in sequenza i programmi relativi alle tabelle da pubblicare.

Il comando è:

```
grrun pgmrun dsv-v-??? [put/get]
```

Il lancio senza parametro [put/get] esegue **entrambe le funzioni**.

La stringa **???** rappresenta la sigla dell'archivio, desumibile in modo completo dal file di mappatura DSVMAP (es: dsv-v-art per gli articoli, dsv-v-mmg per i movimenti di magazzino).

- La direttiva **put** è quella che consente la pubblicazione (o il controllo di coerenza) nella direzione **da Primula a DS**
put è quindi la direttiva da utilizzare per la pubblicazione massiva iniziale.
- La direttiva **get** esegue un controllo **da DS a Primula**; quindi consente di cancellare da DS eventuali record presenti ma non rintracciati nei dati di Primula.

I controlli di tipo get sono molto impegnativi in termini di occupazione delle risorse e di tempi di esecuzione dell'evento da parte del server di pubblicazione. Un unico evento può necessitare di molti secondi o minuti, a seconda del numero di record della tabella lato DS.

Quelli di tipo put, sono altrettanto impegnativi, ma sono "spezzati" in molti eventi, ciascuno dei quali rappresenta decine o centinaia di record (lato Primula).

Non esiste una sequenza prestabilita con cui eseguire la pubblicazione dei dati.

Resta inteso che **vanno altresì rispettati i vincoli di coerenza strutturale o logici.**

Per esempio, essendo la tabella degli articoli strutturata con un codice ID numerico univoco per ogni articolo, **è indispensabile pubblicare prima la tabella articoli di qualsiasi altra tabella che contenga riferimenti agli articoli** (ordini, bolle...).

Così come la relazione fra bolle clienti e fatture. Essendo il numero fattura memorizzato come dato nella tabella delle bolle, occorre pubblicare prima le fatture, poi le bolle. Gli ordini vanno pubblicati prima delle bolle.

Per gli archivi gestiti su più livelli, tipicamente testate e righe, esiste un solo programma di pubblicazione da lanciare sulla testata. Questo evento si occuperà, nel momento in cui viene eseguito, di pubblicare tutti i dati di livello sottostante.

Quindi, per esempio per gli ordini clienti andrà eseguito dsv-v-ord, e questo pubblicherà anche i righe ed i dettagli degli ordini a programma.

Controllo allineamento dati [\[Indice\]](#)

Una importante attività sistemistica di amministrazione del DS è il controllo di **verifica allineamento dei dati**.

Si tratta di rispondere alla domanda se i dati di DS sono identici a quelli di Primula, nelle due direzioni.

Ossia, tutti i dati di Primula siano presenti ed uguali a quelli di DS e – al contrario – nessun dato sia presente in DS ma non presente in Primula (ad eccezione dei dati storicizzati, dove previsto).

La procedura da eseguire è la medesima della pubblicazione massiva iniziale:

```
grrun pgmrun dsv-v-???
```

Ragionevolmente la verifica che si vuole fare sarà in entrambe le direzioni, quindi è probabile che occorra lanciare i comandi **senza direttive put/get** (ossia eseguendo **entrambe** le direttive).

La logica implementata prevede che se il dato di Primula **non risulta presente** nel DS, ci si trova di fronte al caso **riconducibile alla pubblicazione iniziale**; il dato viene inserito in DS.

Se invece il dato è presente in DS, viene fatto un **controllo "carattere-per-carattere"** del record dati di Primula con il record dati di DS.

Con esito positivo (ossia i dati record sono uguali), **non viene fatto nulla**.

In caso di discordanze, chiaramente prevalgono i dati di Primula.

Quindi viene fatta una attività di modifica del record in DS, atta a rendere i dati record uguali nelle due strutture.

La verifica di tipo get, provvede a cancellare da DS eventuali dati record non presenti in Primula.

Tutte le discordanze dati riscontrate, vengono automaticamente loggate nella tabella "log_verifica".

Consultando questa tabella, l'amministratore del database potrà valutare se una attività di riallineamento dei dati possa denotare errori di programmazione o problemi software, che nella maggior parte dei casi necessiteranno interventi di messa a punto dei programmi.

In caso di funzionamento perfetto dei programmi e di stabilità dei sistemi, le attività di controllo allineamento non dovrebbero procurare la scrittura di nessun dato nella tabella log_verifica.

Tabelle di sistema speciali:

Oltre alla tabella di sistema LOG_VERIFICA, illustrata poco sopra, esistono due alte tabelle di sistema importanti:

- **LOG_ERRIO:** vengono inseriti in questa tabella delle informazioni relative ad errori intercorsi nelle procedure eseguite dal server di pubblicazione all'atto dell'esecuzione via DOBC dei comandi SQL
- **LOG:** come già descritto, vengono tracciate tutte le attività di inserimento, modifica, cancellazione eseguite dai vari utenti sulle varie tabelle di DS.

Attenzione: questa tabella potrebbe diventare molto grossa ed arrivare a contenere anche milioni di record; è opportuna una attività sistemistica periodica di svuotamento, al limite selettiva con un'istruzione SQL del tipo:

```
DELETE FROM LOG WHERE DATA < 'aaaa-mm-gg'
```

Casistiche speciali e particolari [\[Indice\]](#)

Articoli

La tabella articoli ha la particolarità di essere identificata come chiave principale tramite un campo ID.

Il codice naturale dell'articolo rappresenta una chiave aggiuntiva secondaria.

Questa scelta è stata fatta – oltre che per similitudine alle logiche consuete (o di moda, a seconda dei punti di vista) con cui vengono creati database gestionali "moderni" – per facilitare le operazioni di **ricodifica** previste dalla nuova procedura rilasciata in rel 7.1

Tutte le tabelle relazionate che fanno riferimento agli articoli, contengono nel loro tracciato non la chiave naturale ma l'ID attribuito automaticamente dal database.

Tabelle/banche

Contrariamente a quanto previsto "nativamente" nei dati gestionali di Primula, anche per queste tabelle è prevista la specificità per azienda.

E' cura dei programmi provvedere al mantenimento contemporaneo delle informazioni inserite, modificate o cancellate in queste tabelle su tutte le aziende attivate all'interno di DS

Movimenti di magazzino

Al fine di consentire una eventuale logica elaborativa esterna che consenta di trattare i movimenti di magazzino in logica "**incrementale**", la tabella dei movimenti di magazzino è stata sviluppata in modalità "solo inserimento".

Detto in altre parole, la chiave principale della tabella è un ID numerico automatico, ed i programmi provvedono solamente ad INSERIRE nuovi record.

All'atto dell'inserimento di un nuovo movimento di magazzino, esso viene come di consueto inserito in DS.

All'atto di una modifica o cancellazione dello stesso movimento, in DS viene **inserito** un nuovo movimento. Se la variazione ha riguardato i campi quantità e/o valore, i numeri riportati nel nuovo movimento saranno **algebricamente compensativi** (es: se ho inserito un movimento di carico con quantità 10, e poi nel gestionale lo modifico portandolo a 8, troverò un nuovo movimento con quantità -2).

Se modifico gli altri dati del record, ad esempio la causale, nel nuovo record inserito troverò il nuovo valore di causale.

Quindi i dati quantità e valore, a parità di chiave naturale del movimento (azienda, deposito, articolo, data e numero movimento) vanno sommati algebricamente; per gli altri campi valgono i valori dell'ID più alto.

Per facilitare la consultazione tradizionale e identica alla logica gestionale di Primula, è presente una vista specifica:

```
SELECT * FROM MOV_MAGAZZINO
```

che ritorna i record "compattati" in una unica voce a parità di chiave naturale.

Archivi "testate/righi"

Gli archivi di tipo testate/righi sono stati generati in DS utilizzando un ID numerico, sia per la testata che per i righi.

Nel record di rigo è indicato l'ID del record di testata a cui fa riferimento.

Le Join SQL per accedere reciprocamente ai dati della tabella relazionata vanno realizzate tramite ID.

Storicizzazioni

Su molte tabelle (es: ordini, bolle, fatture, movimenti contabili) è stato previsto un campo booleano che determina se il record lato gestionale risulta eliminato.

Nelle relazioni testate/righi il flag di **storicizzato** si propaga automaticamente verso il basso.

Backup (dump/vacuum) e restore [\[Indice\]](#)

A prescindere ed indipendentemente dal fatto che le operazioni di backup del filesystem prevedano il salvataggio dell'area dati dell'istanza di database (fatti salvi i problemi potenziali di salvataggio di strutture con tabelle "aperte" e anche la possibilità o meno

di ripristinare tutta la cartella dati del database stesso) è opportuno prevedere una funzione schedulata di **DUMP** dei dati.

Si tratta di lanciare un apposito programma (vedi cartella **SAM\DSV\CMD**)

Questa operazione di backup esegue anche l'importantissima funzione di **vacuum** che serve per il compattamento ed il mantenimento prestazionale del database.

Eseguire queste attività:

- Creare le cartelle di backup **F:\postgres\bck** e **F:\postgres\cmd**
- Copiare il contenuto di SAM\DSV\CMD in F:\postgres\cmd
- Modificare il file **mainbck.cmd**
Settare:
 - BASEDIR: percorso base dei comandi F:\postgres\cmd
 - DIRPGCMD: percorso applicazione Postgres
 - DUMPPDIR: percorso cartella di output dei file dump F:\postgres\bck
 - PGUSER: postgers (utente)
 - PGPASSWORD: [password] dell'utente postgres

E' molto indicato creare dei DUMP di database zippati. Si consiglia quindi di installare il software **7Zip** e di configurare anche:

- FLGZ=yes
- ZIPCMD: percorso e parametri eseguibile 7Zip.exe
- DUMPEXT: estensione file zip

Chiaramente può essere usata qualsiasi altra modalità di zip purché eseguibile da riga comando.

Configurare il comando di lancio, e provare l'esecuzione.

Il comando è già impostato per il DUMP del database con nome "primula". Modificare le due righe "start" in cui vengono lanciati i comandi dbdump.cmd e vacuumdb.cmd per cambiare il nome del db da salvare o aggiungere (max 5) altri nomi database da salvare.

Per quanto riguarda la restore, essa va effettuata – all'occorrenza – con modalità differenti in base al tipo di formato impostato nelle operazioni di backup (dump).

Il formato che è stato impostato nelle procedure di backup distribuite, non consente che l'operazione di restore venga effettuata direttamente dal programma pgAdmin III.

E' quindi necessario utilizzare il comando distribuito "**restore.cmd**".

Logiche di programmazione [\[Indice\]](#)

Non è scopo diretto di questa documentazione analizzare le logiche di dettaglio della programmazione lato Primula di DS, ma solo di dare qualche indicazione di massima.

Per ogni tabella ("**abc**") presente in DS, sono stati creati i seguenti programmi:

- copy **dsvabc.cpy**

Area parametri principale, per la definizione dei campi chiave e per il richiamo di dsv-abc-c.cpy

- **copy dsvabc-c.cpy**

Area parametri di dettaglio contenente tutti i campi del record della tabella di DS; per sua natura è molto simile al tracciato record presente lato cobol (tipo griabc.cpy o griabcrk.cpy)

- **dsv-v-abc.cbl**

Programma per la pubblicazione massiva dei dati della tabella "abc". Questa tipologia di programmi non è necessaria per le tabelle di tipo "collegato" (es: righi), per le quali la pubblicazione massiva viene attivata dalla pubblicazione massiva della tabella principale (es: testata).

- **dsv-p-abc.cbl**

Programma che effettua la lettura del dato da pubblicare in DS nell'archivio di Primula e valorizza tutti i campi del record contenuto in dsv-abc-c.cpy.

Come particolarità abbiamo il fatto che ciascuno di questi programmi è in grado di lanciare **automaticamente (se esistenti)** i programmi dsv-m-abc (Mimosa), dsv-i-abc (Ibisco), **dsv-c-abc (personalizzazioni clienti)**.

Questo significa che in alcuni casi potrà essere sufficiente, **per realizzare una personalizzazione**, scrivere un programma specifico che riceve in input l'area parametri e provveda a valorizzare con qualsiasi regola specifica e customizzata i campi nuovi o esistenti. Nel caso di nuovi campi andranno poi gestiti nelle funzioni SQL di aggiornamento della tabella (dsv-abc.cbl).

- **dsv-abc.cbl**

Programma che esegue analiticamente le funzioni di I/O sul database, sfruttando il meccanismo consentito dal Cobol di attingere alle funzioni **"SQL EMBEDDED"**.

Le funzioni principali previste sono:

- INSERT
- UPDATE
- DELETE
- SELECT

(per acquisizione dati dal record del database e successiva verifica di tipo "put" per controllo coerenza dati)

- **vdsvabc.cbl**

Programma che elabora gli eventi accodati dai vari utenti, relativi ad operazioni di tipo INSERIMENTO, MODIFICA, CANCELLAZIONE eseguite dai vari utenti di Primula, nel corso dell'utilizzo ordinario delle varie funzioni previste dal gestionale.

Per eseguire tali funzioni richiama il programma dsv-abc.cbl

E' predisposto per eseguire le funzioni di verifica di tipo "put" e "get" per il controllo di coerenza dei dati fra gestionale e database.

Nel caso di tabelle strutturate su più livelli (tipo testate/righi), accoda gli eventi di verifica coerenza sull'archivio di livello inferiore ad esso collegato.

- **dsv-f-abc.cbl**

Programma per acquisire via SQL tutti i record della tabella elaborata, e consentire quindi la verifica di coerenza di tipo "get" rispetto ai dati effettivamente presenti nella medesima tabella nel gestionale.

- casi particolari:

- **dsv-idabc.cpy**

Area parametri per consentire la trascodifica fra chiave naturale e ID per le tabelle codificate in DS con chiave principale di tipo ID numerico.

- **dsv-idabc.cbl**

Programma per la lettura tramite la chiave naturale proveniente dal gestionale di un record di una tabella con chiave primaria di tipo ID numerico.

La sequenza SQL utilizzata per questa attività è: creazione e apertura del cursore, select del set di record, fetch dei record (uno alla volta – loop).

Una volta elencati ed esaminati sommariamente i programmi realizzati per la gestione specifica della logica del DS, occorre dire che per i programmi già esistenti sono state previste delle modifiche molto semplici in tutti i punti in cui queste procedure effettuano attività di aggiornamento (write, rewrite, delete) delle tabelle "replicate" nel dataservice.

Questa attività è fondamentale, e rappresenta il punto critico rispetto alla qualità del mantenimento allineato e coordinato dei dati fra gestionale e database.

Infine, ogni tabella creata nel database prevede una codifica tradotta in linguaggio SQL, la quale è possibile riscontrarla nella cartella SAM\DSV\SQL.71.

Esempio: **aggiungere un campo ad una tabella**

- modificare il file dsv-abc-c.cpy, predisponendo all'interno del filler la definizione del nuovo campo
- modificare il programma dsv-p-abc.cbl (oppure in altri casi può essere più opportuno creare un nuovo programma dsv-c-abc).cbl per valorizzare il nuovo campo creato
- modificare il programma dsv-odbc.cbl per provvedere alla gestione SQL del nuovo campo, all'interno delle funzioni INSERT, UPDATE e SELECT
- assicurarsi che il/i programmi che gestiscono la valorizzazione del campo lato Primula, abbiano correttamente previste le chiamate di accodamento dell'evento di pubblicazione sul dataservice
- creare tramite la funzione SQL del programma pgAdmin III il nuovo campo nella tabella, esempio: creare un campo alfanumerico PIC X(30)

```
ALTER TABLE nometabella  
  ADD COLUMN nomecampo character varying(30);  
COMMENT ON COLUMN nometabella.noemcampo IS 'descrizione  
campo';  
UPDATE nometabella SET nomecampo = ' ';
```

- eseguire il comando:

```
grrun pgmrun dsv-v-abc put
```

per provvedere alla valorizzazione del campo nuovo sulla tabella