



APPUNTI IBM i

Youssef Benbouzid

Elmec Informatica (Elmec S.P.A.)



INTRODUZIONE, ACCESSO E COMANDI PRINCIPALI

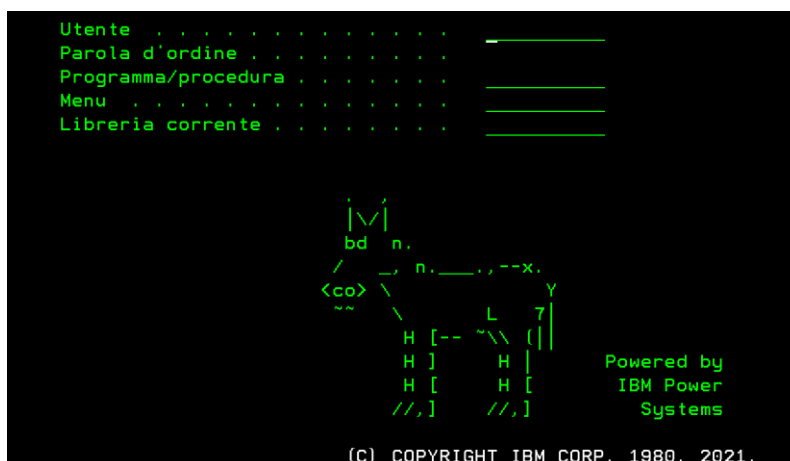
IBM i è il sistema operativo nei server Power Systems (evoluzione di AS/400) di IBM. Il principale linguaggio utilizzato è RPG, al quale si può integrare SQL.

Sono disponibili due strumenti per lo sviluppo IBM i:

- **ACS** (IBM i Access Client Solutions), dal quale si può accedere:
 - Al **Session Manager 5250**, la tradizionale interfaccia-terminale nera-verde;
 - Ad **Esegui script SQL**, che facilita la creazione di query SQL;
- **Rational**, IDE derivato da Eclipse, per gestire file e codice sorgente.

Tenere in mente che la connessione è su 172.16.255.115, che il mio id è elmpben e che non ho disponibile un ELVR. Le credenziali sono elmpben e *****.

Accedere al Session Manager 5250; per la formazione è stata configurata la sessione “AS400 formazione”, avviarla e digitare le credenziali e si apre l’interfaccia (l’”**asinello**”), nel quale digitare nuovamente le credenziali in Utente e Parola d’ordine.



Si accede quindi al menu principale e soprattutto alla linea di comando. È fondamentale alla fine di ogni sessione disconnettersi, digitando dalla linea di comando nel menu principale **90**.

Sono disponibili una serie di comandi selezionabili dalla tastiera con i tasti F; per utilizzarli da portatile bisogna attivare il tasto **fn**; per i tasti superiori all’F12, bisogna fare SHIFT+F(numero che si vuole meno 12, come nell’orologio). In particolare, per tornare indietro si digita **F12**, per far apparire comandi digitati precedentemente **F9**, per aggiornare **F5**.

Da menu principale digitare **strpdm** per accedere al **PDM**, strumento che consente di gestire librerie, oggetti o membri. Digitando **1** si accede alla gestione delle librerie; digitando **ELMP*** compaiono tutte le librerie; altrimenti si digita la propria. Per gestire una libreria si digita **12** nel campo Opz della libreria di interesse.

***L’asinello è stato ora sostituito da un nuovo sistema, connesso a 172.29.255.10 e la sessione è “Nuovo sistema AS400 formazione”.**

OGGETTI SU IBM i

Ogni cosa su IBM i è un oggetto; esistono tre tipi principali di oggetto:

- Librerie (LIBL): contenitori di oggetti; spesso a ogni sviluppatore è associata una libreria (le ultime lettere del nome fanno riferimento al cognome);
- File:
 - Source file (**PF-SRC**): sono “cartelle” che contengono membri;
 - File fisico (**PF**): file che contiene dati;
 - File logico (**LF**): vista logica;
 - Display file (**DSPF**): descrive schermo interattivo nero-verde;
- Programmi (PGM), principalmente di tipo **RPGLE** o **SQLRPGLE**;
- Service Program: modulo condiviso tra programmi.

Le librerie possono contenere gli oggetti dei tipi più svariati (source file, programmi, database, ecc.). Tuttavia, in un ambiente strutturato (come quello per Riva) ogni libreria contiene elementi di tipo omogeneo.

Qui sotto un esempio di disposizione dei file vista da Rational:

```
▼ ELMPBEN
  ▼ ELMPBEN.*lib.test
    > QDSPSRC.*file.pf-src
    ▼ QFILESRC.*file.pf-src
      YBTDLIST.pf
      YBTDLIST01.lf
      YBTDLIST02.lf
    > QRPGLSRC.*file.pf-src
      QSQLSRC.*file.pf-src
    > YBTDLIST.*file.pf-dta
    > YBTDLISTG.*file.pf-dta
      YBTDLIST01.*file.lf
```

SOURCE FILE

Per creare un source file si accede a una libreria (con 12) e si digita sulla linea di comando **crtsrcpf**, convenzionalmente il loro nome inizia con Q e contiene SRC e poi DSP, QSL o RPGLE, dipendentemente da cosa contengono; hanno una lunghezza convenzionale di 92. Per definizione i source file sono multimembro.

Nel definire un source file l'unico altro parametro utile è CCSID (charset), in quanto la codifica di alcuni simboli (come la chiocciola o l'euro) cambia di paese in paese; è quindi utile far riferimento a Unicode (UTF-8 o UTF-16); il *JOB prende il charset 280 (italiano).

FILE FISICI

Si possono creare da Rational, selezionando con tasto destro il source file, poi nuovo membro di tipo PF. Il nome inizia con le prime due lettere del cliente o dello sviluppatore.

I file fisici sono di fatto delle tabelle e interagiscono con **DB2**, il sistema di gestione di database relazionale di IBM; ci si può interfacciare con il linguaggio **DDS** (vecchio stile) o alternativamente con il linguaggio **SQL** (soluzione più moderna). La struttura della tabella prende il nome di “**formato record**”.

Il comando **dspffd file(ELMPBEN/YBTDLIST)** è utile per visualizzare i campi del file fisico.

DDS

DDS è un linguaggio dalla sintassi “arcaica” e fissa in colonne; si può comporre da Rational.

Quello sotto è un esempio di DDS scritto per il file fisico YBTDLIST da Rational. R YBTDLISTR definisce la struttura del record; sotto ci sono i campi con i nomi, i tipi e le lunghezze; ogni nome deve avere al massimo dieci caratteri.

```
.....A.....
000100      A      R YBTDLISTR
000101      A      IDTASKT      10A
000102      A      DESCRIT      100A
000103      A      DATTIMT      Z
000104      A      DONEXX      1A
000105      A      PRIORIT      1S
-----
```

I campi possono essere definiti accedendo alla barra sotto > Programma di richiesta di origine:

Tipo	Campo	Tipo	Posizioni
nome	Nome	Riferimento	lunghezza
▼	PRIORIT	▼	1
		S ▼	

La seguente è la documentazione di IBM che definisce i tipi per file fisici e logici:

<https://www.ibm.com/docs/en/rdfi/9.6.0?topic=44-data-type-physical-logical-files-position-35>

Eventualmente si possono definire campi **varlen**, che allocano puntatori e quindi solo ciò che serve; vanno usati con attenzione, in quanto possono generare buchi durante la fase della cosiddetta riorganizzazione.

Una volta definito, il file fisico va compilato da linea di comando con **crtpf** (da **F4**) e compilando i campi “libreria”, ecc. In seguito per modificare si può digitare il comando **chgpf**.

SQL

Usare SQL è invece un metodo più moderno, si può intervenire sia da 5250, che con lo strumento per script SQL di ACS. SQL interviene sui file fisici, non su quelli logici.

Da 5250 si può digitare il comando **strsql**, e poi digitare le varie query. I campi possono essere definiti su 5250, dopo aver fatto **CREATE TABLE** e **F4**:

Campo	Campo FOR	Tipo	Lunghezza	Scala	Nulli
IDTASKT		CHARACTER	10		3
DESCRIT		CHARACTER	100		3
DATTIMT		TIMESTAMP			3
DONEXT		CHARACTER	1		3
PRIORIT		NUMERIC	1	0	3
					3

Segue...

Esempi di query SQL:

- **CREATE TABLE ELMPBEN/YBTDLIST (IDTASKT CHARACTER (10), DATTIMT TIMESTAMP, PRIORIT NUMERIC (1, 0))**
- **INSERT INTO ELMPBEN/YBTDLIST (IDTASKT) VALUES ('333333333')**
- **SELECT * FROM ELMPBEN/YBTDLIST**

FILE LOGICI

I file logici sono viste logiche associate ai file fisici; non contengono dati, ma fanno riferimento a quelli dei file fisici PF, permettendo di accedervi e filtrarli senza doverli duplicare o modificare.

Si possono creare da Rational come membri di un source file con estensione LF.

```
.....A.....Functions+++++++
000100      A                               UNIQUE
000101      A           R YBTDLISTR         PFILE(YBTDLIST)
000102      A           K IDTASKT
```

Nel file logico qui sopra, **UNIQUE** introduce un vincolo di univocità, R YBTDLISTR fa riferimento al record format YBTDLISTR, **PFILE**(YBTDLIST) specifica il file fisico di riferimento, **K** IDTASKT identifica la chiave primaria; nel fare la insert bisogna quindi tenere in considerazione l'univocità di IDTASKT: quindi, se aggiungo valori duplicati di IDTASKT genero un errore.

Una volta creato il file logico su Rational, si compila su 5250 con **ctrlf** (+ **F4**).

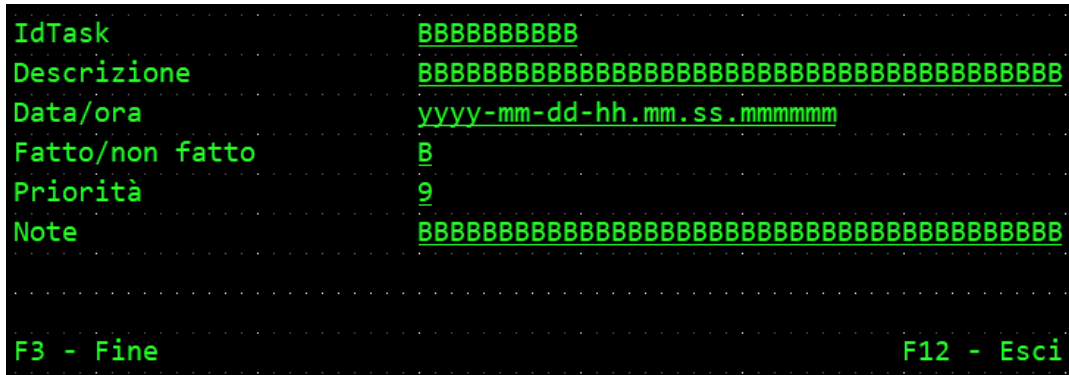
Quando si fa uso di un file logico è necessario che libreria sia nella lista delle librerie.

Quando viene modificato un file logico o display ma non viene ricompilato l'RPG si ha un **errore di livello**; per verificare è disponibile il comando **dspobjd** con nome del programma. Per mostrare dettagli degli errori è disponibile il comando **wrksplf**.

DISPLAY FILE

I display file definiscono il layout della schermata nera-verde e i relativi campi.

Si possono generare da Rational come membri di estensione DSPF, componendoli in modalità drag-and-drop attraverso lo strumento DDS Design di Rational.



IdTask	BBBBBBBBBB
Descrizione	BB
Data/ora	yyyy-mm-dd-hh.mm.ss.mmmmmmm
Fatto/non fatto	B
Priorità	9
Note	BB
F3 - Fine	
F12 - Esci	

Lo schermo nero è costituito da uno **standard record** e viene chiamato WIN01 (nome settabile da design o da origine). Si può strutturare il display file con un subfile record WCTL1 (contenitore pagina e controllo), uno standard record WSFL1 e un WFOOTER (per i function keys).

Successivamente si aggiungono a sinistra i **Text Constant** con i vari campi, come la data, ecc. Per dare un significato ai Text Constant si aggiungono delle keyword.

Si aggiungono successivamente le cosiddette **Function Keys**, che aggiungono funzionalità interattive alla videata, del tipo “F12-Esci”, le quali possono essere di tipo **command attention** o **command function**. F12-esci viene indicato ad esempio con KL. Di seguito la documentazione relativa: <https://www.ibm.com/docs/en/i/7.3.0?topic=indicators-function-key>

A destra si aggiungono i **Named Field** (assimilabili a variabili), da chiamare ad esempio W1DATTIMT, settabili come input o output, a cui attribuire tipo e lunghezza e che possono assumere i valori delle keyword.

Il display file si compila attraverso il comando **crtdspf** seguito da **F4**.

FILE RPGLE E SQLRPGLE

Il linguaggio RPG viene utilizzato nell’ultima versione IV (ILE RPG), la quale introduce aspetti moderni nella programmazione. La distinzione tra i file sorgente di tipo **rpgle** e **sqlrpgle** (generabili come membri da Rational con le omonime estensioni) sta nei differenti standard nelle query SQL:

- In RPGLE: si usano i comandi chain, write, read e update, operando sui file fisici e logici. Si compila il file con **crtbndrpg** con F4
- In SQLRPGLE: si usano le istruzioni SQL classiche, come SELECT, INSERT, UPDATE e DELETE. Si compila con **crtsqlrpgi**.

LINGUAGGIO RPG

Un programma RPG è diviso in blocchi; nella prima riga si digita ****free**.

Le specifiche iniziali sono dei seguenti tipi:

- **H**: specifiche di tipo include e debug;
- **F**: file da utilizzare nel programma;
- **D**: variabili globali;
- **C**: istruzioni;
- **ctl-opt**: dichiarazione delle specifiche iniziali;
- **dcl-f**: dichiarazione di file fisici e logici;
- **dcl-s**: dichiarazione di variabili.

Si possono poi realizzare moduli riutilizzabili di codice, di due tipi:

- **subroutine**: pezzi di codice senza parametri o variabili locali. Incluse tra **begsr** ed **endsr**;
- **procedure**: con variabili locali e parametri, usate anche all'interno del service program. Incluse tra **dcl-proc** ed **end-proc**; i parametri (che possono essere ritornati con return) sono tra **dcl-pi** ed **end-pi**.

All'interno del codice sorgente l'istruzione **return** è necessaria per la compilazione.

```
dcl-proc Procedura;  
  dcl-pi *n ind;  
    idTask_inp char(10);  
    dattim_inp timestamp;  
    priori_inp zoned(1:0);  
  END-PI;  
END-PROC;  
  
*inLR = *on;  
return;
```

Il comando **call pgm(nomefilerpg)** permette di avviare il programma, **strdbg nomeprogramma** si usa per debuggare.

All'istruzione **chain** si può passare tra parentesi una chiave che identifica univocamente un record. In SQL la chain viene sostituita da **SELECT**. **chain(n)** si usa solo in lettura, in quanto locka il record.

Per inserire istruzioni SQL in un programma RPG (in particolare sqlrpgle) si scrive l'istruzione **EXEC SQL** al loro inizio.

La **commit** consiste in un controllo di sincronia; se questa è attiva, dopo ogni istruzione è necessario effettuare una commit, altrimenti tutto ciò che è stato effettuato è transitorio; per ovviare a questo problema si può digitare nel codice del programma l'istruzione **EXEC SQL SET OPTION COMMIT=*NONE**.

SPECIFICHE INIZIALI

Di seguito sono mostrati frammenti di un file RPG per la gestione dei record. Le seguenti sono le specifiche iniziali:

```
**free
dcl-f VISTALOGICA keyed usage(*input:*output:*update:*delete);
dcl-s msg char(20); // Variabile stringa di venti caratteri
dcl-s i int(3); // Variabile intera di tre cifre
```

SCRITTURA RECORD

Il seguente è il codice per la scrittura nativa di un record:

```
// Scrittura nativa
STATOF = '1';
SOCIEF = '01';
MAGAZF = '02';
COPERF = newOpe;
DESCRF = newDesc;
ATTIVF = '';
// Necessario gestire gli errori
write(e) FORMATORECORD;
if %error();
    // Gestione errore
    esito = 'E';
ENDIF;
```

Si include la vista logica opportuna con:

dcl-f VISTALOGICA keyed usage(*input:*output:*update:*delete)

Il seguente è il codice corrispondente in SQL:

```
// Scrittura con SQL
EXEC SQL INSERT INTO TCAFRMPF(STATOF, SOCIEF, MAGAZF, COPERF, DESCRF, ATTIVF)
    VALUES ('1', '01', '02', :NEWoPE, :NEWdESC, '');
// Controllo che non ci siano errori e che i record non siano finiti
if sqlCode >= 0 and sqlCode <> 100;
    // Inserimento avvenuto con successo
    esito = '';
else;
    esito = 'E';
ENDIF;
```

In entrambi i casi aggiungo un record solo.

MODIFICA RECORD

Il seguente è il codice per la modifica nativa di un record:

```
// Modifica nativa
STATOF = '1';
SOCIEF = '01';
MAGAZF = '02';
COPERF = 'XXXXX';
ATTIVF = '';
chain (STATOF:SOCIEF:MAGAZF:COPERF) VISTALOGICA;
if %found();
    // Necessario gestire gli errori
    update(e) VISTALOGICA;
    // Il trim è necessario per togliere spazi da campo originario
    DESCRF = %trim(DESCRF) + ' ' + 'modificato'
    if %error();
        // Gestione errore
        esito = 'E';
    ENDIF;
ENDIF;
```

Di seguito la documentazione relativa: <https://www.ibm.com/docs/it/i/7.5.0?topic=codes-update-modify-existing-record>

Il seguente è il corrispondente con SQL:

```
// Modifica con SQL
EXEC SQL UPDATE TCAFRMPF SET DESCRF = trim(DESCRF) concat '(modificato sql)', ATTIVF = 'N'
    WHERE STATOF = '1' and SOCIEF = '01' and MAGAZF = '02' and COPERF = 'XXXXX';
// Controllo che non ci siano errori e che i record non siano finiti
if sqlCode >= 0 and sqlCode <> 100;
    // Modifica avvenuta con successo
    esito = '';
else;
    esito = 'E';
ENDIF;
```

La chain mi permette di posizionarmi sul record di interesse. La chiave deve corrispondere alla vista logica utilizzata. La chain, in assenza di informazioni precise, mi potrebbe posizionare nel primo record coerente alle informazioni fornite.

Per selezionare tutti i record si può ricorrere a un ciclo.

CANCELLAZIONE RECORD

Il seguente è il codice per la cancellazione nativa:

```
// Cancellazione nativa
STATOF = '1';
SOCIEF = '01';
MAGAZF = '02';
COPERF = 'XXXXX';
chain (STATOF:SOCIEF:MAGAZF:COPERF) VISTALOGICA;
if %found();
    // Necessario gestire gli errori
    delete(e) VISTALOGICA;
    if %error();
        // Gestione errore
        esito = 'E';
    ENDIF;
ENDIF;
```

Qui di seguito la documentazione relativa: <https://www.ibm.com/docs/it/i/7.5.0?topic=codes-delete-delete-record>

E qui il codice SQL corrispondente:

```
// Cancellazione con SQL
EXEC SQL DELETE FROM TCAFRMPF
        WHERE STATOF = '1' and SOCIEF = '01' and MAGAZF = '02' and COPERF = 'XXXXX';
// Controllo che non ci siano errori e che i record non siano finiti
if sqlCode >= 0 and sqlCode <> 100;
    // Cancellazione avvenuta con successo
    esito = '';
else;
    esito = 'E';
ENDIF;
```

LETTURA RECORD

```
// Lettura nativa di un singolo record
STATOF = '3';
SOCIEF = '01';
MAGAZF = '01';
COPERF = '38530';
chain(STATOF:SOCIEF:MAGAZF:COPERF) VISTALOGICA;
if %found();
    // Record trovato
ENDIF;

// Lettura di un singolo record con SQL
EXEC SQL SELECT DESCRF, ATTIVF
        INTO :dsOpe
        FROM VISTALOGICA
        WHERE STATOF = '3' and SOCIEF = '01' and MAGAZF = '01' and COPERF = '38530';
```

In particolare questo frammento di codice fa riferimento alla struttura dati dsOpe:

```
dcl-ds dsOpe qualified;
    descrizione char(280);
    attivo char(1);
END-DS;

// Lettura nativa di più record
STATOF = '2';
SOCIEF = '06';
MAGAZF = '07';
setll (STATOF:SOCIEF:MAGAZF) VISTALOGICA;
reade (STATOF:SOCIEF:MAGAZF) VISTALOGICA;
doU %eof(VISTALOGICA);
    dsOpe.descrizione = DESCRF;
    dsOpe.attivo = ATTIVF;
    reade (STATOF:SOCIEF:MAGAZF) VISTALOGICA;
ENDDO;

// Lettura di più record con SQL
EXEC SQL DECLARE Cur01 CURSOR FOR
        SELECT descrf, attivf
        FROM VISTALOGICA
        WHERE STATOF = '2' and SOCIEF = '06' and MAGAZF = '07'
        FOR READ ONLY
```

Il cursore è un puntatore che va a scorrere il set di dati.

Con setll mi posiziono nel record di interesse e poi interviene reade.

Setll *loyal scorre tutto, seguira da read VISTALOGICA

GESTIONE DISPLAY

Questo invece è un frammento di un file rpgle per la gestione di un display file:

```
000125 dcl-proc VisualizzaTask;
000126     //dcl-pi *n char(10); Equivarrebbe a char VisualizzaTask
000127     dcl-pi *n; // Equivale a void VisualizzaTask
000128     idTask_inp char(10); // Equivale a void VisualizzaTask (char idTask)
000129     END-PI;
000130     pulisciCampi();
000131     chain(n) (idTask_inp) YBTDLIST01;
000132     if %found();
000133         W1IDTASK = IDTASKT;
000134         W1DESCRIT = DESCRIT;
000135         W1DATTIMT = DATTIMT;
000136         W1DONEXX = DONEXX;
000137         W1PRIORIT = PRIORIT;
000138         W1NOTEXX = NOTEXX;
000139     ENDIF;
000140 END-PROC;
```

Questo è il frammento di un file sqlrpgle per la gestione del display file:

```
000128 dcl-proc VisualizzaTask;
000129     dcl-pi *n;
000130     idTask_inp char(10);
000131     end-pi;
000135     pulisciCampi();
000136     W1IDTASK = IDTASKT;
000137     W1DESCRI = DESCRIT;
000138     W1DATTIM = DATTIMT;
000139     W1DONEXX = DONEXXT;
000140     W1PRIORI = PRIORIT;
000141     W1NOTEXX = NOTEXXT;
000142     EXEC SQL // Inizio istruzioni SQL
000143     SELECT IDTASKT, DESCRIT, DATTIMT, DONNEXT, PRIORIT, NOTEXXT // Seleziono campi
000144     INTO :W1IDTASK, W1DESCRI, W1DATTIM, W1DONNEX, W1PRIORI, W1NOTEXX // Metto nei campi display
000145     FROM YBTDLIST // Campi da questa tabella
000146     WHERE IDTASKT = :idTask_inp;
000147 end-proc;
```

Si può realizzare una **struttura dati** chiamata dsTask per evitare di dover riscrivere i campi:

```
000151     dcl-ds dsTask qualified;
000152         id char(10);
000153         descr char(100);
000154         data timestamp;
000155         done char(1);
000156         prior zoned(1);
000157         note varchar(100);
000158     end-ds;
000159     EXEC SQL // Inizio istruzioni SQL
000160     SELECT IDTASKT, DESCRIT, DATTIMT, DONNEXT, PRIORIT, NOTEXXT // Seleziono campi
000161     INTO :dsTask
```

Questo è un esempio di codice che permette di selezionare varie opzioni:

```
doU *inKL = *on;
EXFMT WIN01;
if w1idTask = '';
    *in20 = *off;
else;
    *in20 = *off;
ENDIF;
if w1descri = '';
    *in22 = *off;
else;
    *in22 = *off;
ENDIF;
select;
when *inKF = *on;
    esito = NuovoTask(W1IdTask:W1Descri:W1Dattim:W1Donexx:W1Priori);
    if esito = *on; //fai cose
    ENDIF;
when *inKB = *on;
    esito = ModificaTask(W1IdTask:W1Descri:W1Dattim:W1Donexx:W1Priori);
    if esito = *on; //fai cose
    ENDIF;
when *inKQ = *on;
    esito = EliminaTaskTask(W1IdTask);
    if esito = *on; //fai cose
    ENDIF;
other;
ENDSL;
ENDDO;
```

Questa è una lettura con **setll** e **reade**:

```
// setll punta, ma non legge, deve seguirlo reade
setll (userint) VISTALOGICA;
// reade significa read con chiave equivalente
reade (userint) VISTALOGICA;
doU %eof(VISTALOGICA);
    dsply idtaskt;
    // In assenza del seguente reade, non andrei avanti nel ciclo
    reade (userint) VISTALOGICA;
enddo;
```

DSPJOB

Il comando **dspjob** permette di visualizzare informazioni dettagliate di un job (lavoro corrente), in particolare:

- **Spool**: archivi temporanei dell'output; con il comando **wrksplf** si possono visualizzare gli spool di compilazione (e vedere gli errori che impediscono la compilazione); se incollo il codice dell'errore su "Ricerca" e faccio **F16** posso vedere cosa non va
- **DSPJOBLOG**: comando utile per il debug e per identificare quindi gli errori in un job; **F10** per visualizzare gli errori;
- **Stem**: la sequenza di tutti i programmi chiamati da un programma RPG;
- **Lista librerie**: per accedere alle librerie.

Approfondimento sulla lista librerie: nel sistema informativo le librerie sono strutturate secondo un indice di priorità, al quale si può fare riferimento con ***LIBL**. Per mostrare la lista librerie è disponibile il comando **dsplibl**, per gestirle **edtlibl**. In Elmec abbiamo MBE Elmec per comporre la lista librerie. **Aggiungo la mia libreria alla lista delle librerie ogni volta che devo compilare un file RPG che include altri file.**

SUBFILE

Un **subfile** rappresenta una componente del display file ed è costituito da una lista di righe visualizzate su 5250 per mostrare i record, per poterli scorrere e selezionare. I subfile vengono controllati dai **control file**.

SERVICE PROGRAM

Un **service program** rappresenta il corrispettivo di una libreria nel mondo IBM i/RPG. Costituisce un insieme di procedure accessibili da più programmi RPG. Best practice metterlo in source file a parte QSRVPGMSRC.

L'istruzione **/COPY** in un programma RPG, chiamata "**barra copy**", importa definizioni (strutture dati, procedure, ecc.) da altri file. È assimilabile a #include di C.

La firma rappresenta identificativo del service program e si scrive con linguaggio particolare. Best practice metterla in source file QBNDSRC. Non ha bisogno di essere compilata.

Un service program si compila come un normale file RPG, mettendo come "Tipo compilazione" "MODULE" al posto di PGM. Il service program può quindi essere utilizzato digitando **crtsrvgpm + F4**, precisando il nome firma e relativo source file.

PROFOUND UI

Profound UI è una piattaforma low code per la creazione di interfacce utente moderne alternative alle tradizionali schermate nero-verdi.

RIPASSO ED ESEMPIO PRATICO DI SVILUPPO DI UN'APPLICAZIONE

Realizziamo un esempio di applicazione in IBM i che consente di gestire un elenco di libri. La sviluppiamo con RPG ed SQL e includiamo un subfile e service.

SOURCE FILE

Il primo passaggio consiste nella realizzazione dei source file, che possano contenere i nostri file ("membri"). Si accede a 5250 e tramite il comando CRTSRCPF si creano i source file QFILESRC (per file fisici ed eventuali file logici), QRPGLSRC (per file RPGLE ed SQLRPGLE), QDSPFSRC (per i display), QSRVPGMSRC (per i service program).

FILE FISICO

Accedere alla propria libreria su Rational, selezionare col tast destro il source file QFILESRC e creare un file fisico YBLIBRI di estensione pf.

000100	A	R YBLIBRIR	
000101	A	IDLIBRO	10A
000102	A	AUTORE	10A
000103	A	TITOLO	15A
000104	A	ANNO	4A
000105	A	ISBN	13A
000106	A	DISP	2A
000107	A	NUMERO	1S 0

Nell'immagine sopra è indicato il "formato record", scritto in linguaggio DDS. La prima riga definisce il formato record (ossia i campi della tabella). Le righe seguenti definiscono le variabili e relativi tipi (A ad esempio sta per "carattere", qui la documentazione:

<https://www.ibm.com/docs/en/rdfi/9.6.0?topic=44-data-type-physical-logical-files-position-35>) e lunghezze.

FILE LOGICO

Dal momento che abbiamo optato di realizzare l'applicazione usando SQL, non serve realizzare un file logico (che si utilizza nel caso si optasse per le istruzioni native invece di SQL).

FILE DISPLAY

Il display lo realizziamo con subfile (e relativo control file). Il subfile costituisce un elenco sul quale un utente può intervenire, il control file permette di controllarlo.

Quindi creare un file YBDLIBRI sul source file relativo da rational con estensione DSPF.

Generalmente di preferisce fare uso dello strumento no-code reso disponibile da Rational, invece che fare uso solo di DDS.

Quindi, dal menu a destra, trascinare il subfile ("Record file secondario", che genera anche control file). Chiamare quindi il subfile WSFL e il control file WCTL (best practice).

Nel control file trascinare dei “costanti di testo” e chiamarli con i nomi dei campi (idLibro, eccetera). In seguito, trascinare “campo denominato” e definirne nomi (WIDLIBRO, eccetera), tipi (carattere, eccetera), lunghezze, input/output. **Aggiungiamo poi un campo WSEL per poter scorrere il subfile.**

Aggiungiamo dei tasti funzione, per le funzioni “AGGIUNGI”, “MODIFICA”, “CANCELLA”, associate a dei tasti F. Aggiungere quindi relativi costanti di testo e in seguito i tasti funzione veri e propri, cliccare sul display, accedere a “Tasti funzione” e selezionare F1, F2, F3, F12 con “Attenzione comando”.

Questo è il nostro control file:

F1 - Aggiungi				F2 - Modifica				F3 - Cancella			
idLibro	Autore	Titolo	Anno	ISBN	Disp	Numero					
BBBBBBBBBB	BBBBBBBBBB	BBBBBBBBBBBBBBBB	BBBB	BBBBBBBBBBBBBBBB	00	9	B				
BBBBBBBBBB	BBBBBBBBBB	BBBBBBBBBBBBBBBB	BBBB	BBBBBBBBBBBBBBBB	00	9	B				
BBBBBBBBBB	BBBBBBBBBB	BBBBBBBBBBBBBBBB	BBBB	BBBBBBBBBBBBBBBB	00	9	B				
BBBBBBBBBB	BBBBBBBBBB	BBBBBBBBBBBBBBBB	BBBB	BBBBBBBBBBBBBBBB	00	9	B				
BBBBBBBBBB	BBBBBBBBBB	BBBBBBBBBBBBBBBB	BBBB	BBBBBBBBBBBBBBBB	00	9	B				
BBBBBBBBBB	BBBBBBBBBB	BBBBBBBBBBBBBBBB	BBBB	BBBBBBBBBBBBBBBB	00	9	B				

Intervenire ora sul DDS (si accede da “Origine”), associare 51 a SFLDSPCTL, 52 a SFLDSP e 50 a SFLCLR e SFLDLT (per cancellare il subfile e caricarne poi uno nuovo). Aggiungere SF1SRN 4 H SFLRCDNBR (serve a scorrere il subfile).

Aggiungere un WFOOTER (un “Record standard”) con testo costante “F12 – Esci”, per consentire all’utente di uscire.

Ricordiamoci che F1, F2, F3, F12 corrispondono a KA, KB, KC, KL (qui la documentazione: <https://www.ibm.com/docs/en/i/7.3.0?topic=indicators-function-key>).

Il seguente è il nostro DDS del display file:

```

000001      A          R WSFL                      SFL
000002      A          WIDLIBRO          10  0  5  3
000003      A          WAUTORE          10  0  5 14
000004      A          WTITOLO          15  0  5 25
000005      A          WANNNO           4  0  5 41
000006      A          WISBN            13  0  5 46
000007      A          WDISP             2  0  5 61
000008      A          WNUMERO          15  0  5 67
000009      A          WSEL              1  8  5 75
000010      A          R WCTL                      SFLCTL(WSFL)
000011      A          51                      SFLDSPCTL
000012      A          52                      SFLDSP
000013      A          50                      SFLCLR
000014      A          50                      SFLDLT
000015      A                      SFLPAG(6)
000016      A                      SFLSIZ(12)
000018      A                      CA01
000019      A                      CA02
000020      A                      CA03
000021      A                      CA12
000022      A                      OVERLAY
000023      A          SF1SRN          4  0H  SFLRCDNBR
000024      A                      4 14'Autore'
000025      A                      4 25'Titolo'
000026      A                      4 41'Anno'
000027      A                      4 46'ISBN'
000028      A                      4 60'Disp'
000029      A                      4 65'Numero'
000030      A                      4  3'idLibro'
000031      A                      2 34'F2 - Modifica'
000032      A                      2 55'F3 - Cancella'
000033      A                      2 13'F1 - Aggiungi'
000035      A          R WFOOTER
000036      A                      21 51'F12 - Esci'

```

FILE RPG

Creare un file YBRLIBRI sul relativo source file di estensione SQLRPGLE (dato che lavoriamo anche con SQL).

Le seguenti sono le specifiche iniziali; *free indica che usiamo il formato free, per introdurre la sintassi moderna:

```

000100 **free
000101 ctl-opt dftactgrp(*no); // Il programma gira in ILE
000102 dcl-f YBRLIBRI WORKSTN(*EXT) SFILE(WSFL:SF1SRN); // Includo display e subfile
000112
000113 EXEC SQL SET OPTION COMMIT=*NONE; // Per disabilitare il commit
000114
000115 doU *inKL = *on; // Finché non viene selezionato KL (F12)
000116   PulisciWSFL();
000117   CaricaWSFL();
000118   *in51 = *on; // Abilita SFLDSPCTL (controllo subfile)
000119   if SF1SRN > 0; // Se ci sono record in subfile
000120     *in52 = *on; // Abilita SFLDSP (visualizzazione subfile)
000121     SF1SRN = 1; // Cursore nel primo record
000122   ENDIF;
000123   VisualizzaWSFL();
000124 ENDDO;
000130
000131 *inLr = *on; // Termina il programma alla fine
000132 return; // Per compilare file RPG

```

Le seguenti sono procedure relative al subfile: pulizia, caricamento, visualizzazione:

```
000114 //-----
000115 // Pulizia subfile
000116 //-----
000117 dcl-proc PulisciWSFL;
000118 *in50 = *on; // Abilita SFLCLR e SFLDLT (cancellazione)
000119 *in51 = *off; // Disabilita SFLDSPCTL (controllo subfile)
000120 *in52 = *off; // Disabilita SFLDSP (visualizzazione subfile)
000121 write WCTL; // Riscrive control file
000122 *in50 = *off; // Disabilita la cancellazione
000123 SF1SRN = 0; // Record number da capo
000124 END-PROC;

000126 //-----
000127 // Caricamento subfile
000128 //-----
000130 dcl-proc CaricaWSFL;
000131 // Struttura dati coerente al file fisico
000135 dcl-ds dsLibro qualified; // qualified abilita prefisso dsLibro
000136 idLibro char(10);
000137 autore char(10);
000138 titolo char(15);
000139 anno char(4);
000140 isbn char(13);
000141 disp char(2);
000142 numero zoned(1);
000151 END-DS;
000155
000156 // Prelevo i campi dal file fisico
000157 EXEC SQL
000158 DECLARE C1 CURSOR FOR
000159 SELECT IDLIBRO, AUTORE, TITOLO, ANNO, ISBN, DISP, NUMERO
000160 FROM YBLIBRI;
000162
000163 EXEC SQL OPEN C1;
000164 dow sqlcod = 0 and sqlcod <> 100;
000165 EXEC SQL FETCH C1 INTO :dsLibro;
000166 if sqlcod < 0 or sqlcod = 100;
000167 leave;
000168 ENDIF;
000169 WIDLIBRO = dsLibro.idLibro;
000170 WAUTORE = dsLibro.autore;
000171 WTITOLO = dsLibro.titolo;
000172 WANNO = dsLibro.anno;
000173 WISBN = dsLibro.isbn;
000174 WDISP = dsLibro.disp;
000175 WNUMERO = dsLibro.numero;
000176 SF1SRN +=1; // Incremento il record number
000177 write(e) WSFL; // Scrivo nel subfile
000178 ENDDO;
000179 EXEC SQL CLOSE C1;
000194 END-PROC;

000196 //-----
000197 // Visualizzazione subfile
000198 //-----
000199 dcl-proc VisualizzaWSFL;
000206 write WFOOTER; // Aggiunge il footer
000207 exfmt WCTL; // Visualizza subfile e accetta input nel control file
000209 END-PROC;
```

RISULTATO

F1 - Aggiungi			F2 - Modifica		F3 - Cancella		
idLibro	Autore	Titolo	Anno	ISBN	Disp	Numero	
0000000001	Rowling	Harry Potter	1997	9780000000000	si	9	—
0000000002	Malika	Libro cuore	1999	9780000000001	no	0	—
0000000003	Chicco	Cucina lumbarda	2004	9789898955555	no	0	—
0000000005	Fitzgerald	Il grande	1925	9780000000004	SI	6	—
0000000009	Tolkien	Signore Anelli	1954	9780000000001	SI	8	—
0000000008	Dostoevsky	Delitto	1866	9780000000007	SI	3	—
F12 - Esci							

COMPILAZIONE E AVVIO

Accedere al 5250, digitare il comando EDTLIBL e aggiungere quindi la propria libreria alla lista delle librerie (questo ogni volta).

Ora carichiamo la tabella digitando STRSQL. Per usare SQL si può usare ACS alternativamente a 5250.

Alternativamente alla creazione del file fisico si può usare direttamente SQL per creare la tabella; su 5250 è CREATE TABLE + F4; poi la query:

- CREATE TABLE ELMPBEN/YBLIBRI(IDLIBRO CHARACTER(10), AUTORE CHARACTER(10), TITOLO CHARACTER(10), ANNO CHARACTER(4), ISBN CHARACTER(10), DISP CHARACTER(2), NUMERO NUMERIC(1, 0))

Comunque, per inserire valori nella tabella:

- INSERT INTO ELMPBEN/YBLIBRI(IDLIBRO, AUTORE, TITOLO, ANNO, ISBN, DISP, NUMERO) VALUES('0000000001', 'ROWLING', 'HARRY POTTER', '1997', '9780000000000', 'SI', 9)

Per selezionare campi:

- SELECT * FROM ELMPBEN/YBLIBRI

Per compilare digitare il tutto:

- CRTPF + F4 + specifiche (compila file fisico); **questo prima delle query insert.**
- CRTDSPF + F4 + specifiche (compila display);
- CRTSQLRPGI + F4 + specifiche (compila RPG);
- CALL PGM + F4 + specifiche (esegue RPG).

AMPLIAMENTO DELL'APPLICAZIONE CON SERVICE PROGRAM

BARRA COPY

Le barra copy costituiscono file RPG che raccolgono porzioni di codice utilizzate da più programmi (principalmente strutture dati) durante la compilazione; si crea un source file QCOPYSRC a esse riservato. Come convenzione si usa la I nel nome (per “include”). Quindi YBILIBRI.

Le barra copy le facciamo partire dalla colonna 7 perché potrebbero essere richiamate da programmi che usano la sintassi posizionale:

```
RR7      dcl-ds tDsParam qualified template;
000103      optype char(1);
000104      idLibro char(10);
RR7      END-DS;
```

FIRMA

Una firma è un file di estensione .bnd a cui viene riservato un source file QBNDSRC, scritto con un linguaggio a parte, che definisce le procedure del service program.

```
000100      STRPGMEXP PGMLVL(*CURRENT) SIGNATURE('YBSPROG.001')
000101      EXPORT      SYMBOL('YB_AGGIUNGILIBRO')
000102      EXPORT      SYMBOL('YB_MODIFICILIBRO')
000103      EXPORT      SYMBOL('YB_CANCELLALIBRO')
000104      EXPORT      SYMBOL('YB_OTTIENILIBRO')
000105      ENDPGMEXP
```

SERVICE PROGRAM

Un service program raccoglie procedure chiamate da altri programmi durante la compilazione. Il service program viene compilato una volta e poi linkato.

```
000100 **free
000101 Ctl-Opt nomain;
000102
000103 /COPY ELMPBEN/QCOPYSRC,YBILIBRO // Inclusionione della barra copy
000104 EXEC SQL SET OPTION COMMIT = *none;
000105
000107 //-----
000108 // Aggiungi libro
000109 //-----
000110 dcl-proc AggiungiLibro export; // Per esportare la procedura
```

Per generare e compilare un service program:

- Per compilare un service program sono necessari una barra copy e una firma;
- Si crea un source file QSRVPGMSRC a essi riservato. Si compilano come normali programmi RPG (quindi con CRTSQLRPGI + F4 ad esempio), mettendo *MODULE in “Tipo compilazione” al posto di *PGM; ci si ritrova un oggetto compilato ma non ancora eseguibile;
- Il prossimo passaggio è CRTSRVPGM + F4, mettendo le specifiche della firma.

```

Service program . . . . . ybsprog      Name
Library . . . . . elmpben      Name, *CURLIB
Module . . . . . ybsprog      Name, generic*, *SRVPGM, *ALL
Library . . . . . elmpben      Name, *LIBL, *CURLIB...
+ for more values

Export . . . . . *SRCFILE      *SRCFILE, *ALL
Export source file . . . . . QbndSRC      Name, QSRVSR
Library . . . . . elmpben      Name, *LIBL, *CURLIB
Export source member . . . . . ybsprog      Name, *SRVPGM
Export source stream file . . .

```

Per collegare un programma a un service program:

- CRTBNDDIR + F4 crea YBBLIBRI;
- WRKBNDDIRE + F4 per gestire il binddir;

```

-----Creation-----
Opt  Object      Type      Library      Activation  Date      Time
  1   ybsprog     *SRVPGM    *LIBL        *IMMED
(No binding directory entries for this binding directory.)

```

- Vi si fa riferimento nel programma con bnddir(YBBLIBRI).

BEST PRACTICE E SUGGERIMENTI

Buone pratiche:

- Chiamare un source file con iniziale Q e rendere chiaro nel nome cosa contengono;
- I nomi dei membri iniziano con le prime due iniziali del cliente (o del nome e cognome dello sviluppatore), seguite da prima lettera del tipo del file; ad esempio: YBLIBRI (file fisico), YBRLIBRI (file RPG), YBDLIBRI (file display);
- Se si blocca il terminale di 5250 mentre si scrive un comando digitare Ctrl;
- Alcune tastiere non hanno tasti F maggiori all'F12; digitare SHIFT e ragionare come per l'orologio; ad esempio, F20 = SHIFT + F8.

Per gestire errori di compilazione:

- DSPJOBLOG per visionare gli errori;
- WRKSPLF + 5 (in libreria) + B (in CONTROL) per visionare la natura degli errori; F19 per spostare a sinistra e F20 a destra;
- WRKJOB per vedere se un file è presente;
- STRDBG per il debug.
- EV DSENTRY per spaccettare una struttura passata come parametro.