IBM i

SQL + RPG x IFS

leggere e scrivere stream files con RPG e SQL

Marco Riva

MK1

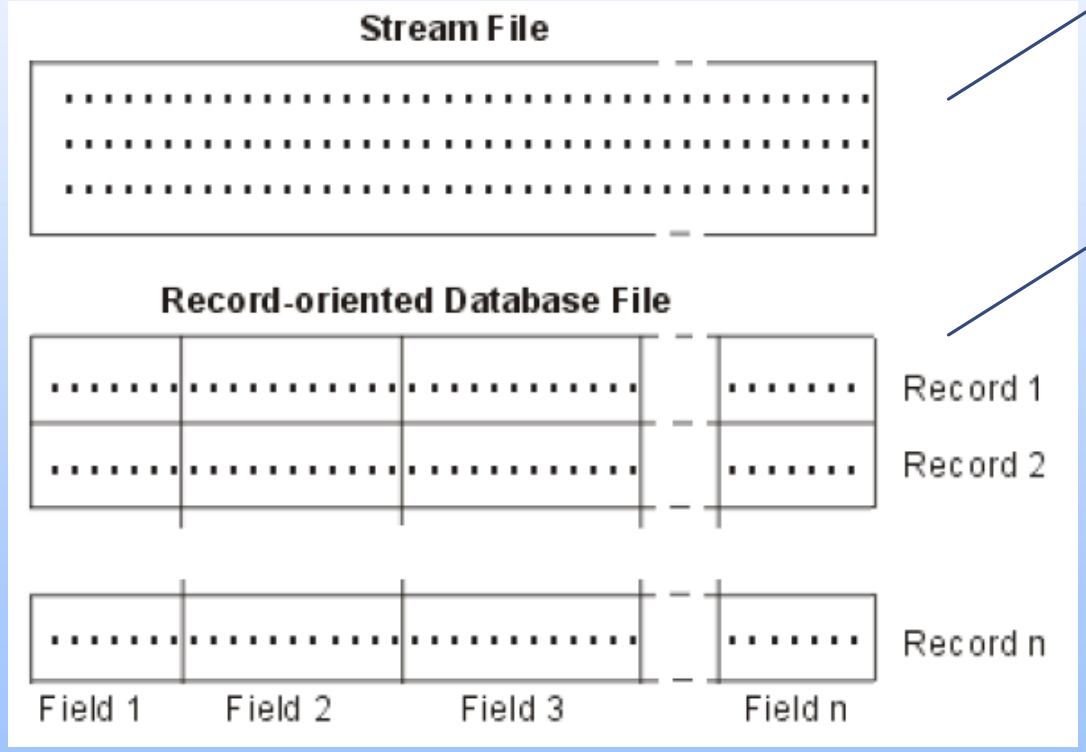
www.markonetools.it

IBMCHAMPION 2021 Power Systems area

Ultimo aggiornamento: 17/07/2021

Stream file vs. record file

- Uno **stream file** è una sequenza di byte accessibile in maniera casuale, con nessuna ulteriore struttura imposta dal sistema



IFS

LIBRERIE

LOB

Large Object

- LOB è l'acronimo generico per fare riferimento a tipi dati
  - **BLOB**: binary large data objects
  - **CLOB**: single-byte character data objects
  - **DBCLOB**: double-byte character data objects
- In questo tutorial ci occuperemo di **CLOB**

LOB file reference variables

- Sono variabili host che consentono di trasferire dati da e verso stream files memorizzati su IFS
- Rappresenta (piuttosto che contiene) il file
- Una variabile file reference ha tipo dati BLOB_FILE, CLOB_FILE o DBCLOB_FILE
- https://www.ibm.com/docs/en/i/7.4?topic=dhviiratus-declaring-lob-host-variables-in-ile-rpg-applications-that-use-sql

Tipi dati SQL – LOB file reference

- **clob_file** → in RPG è rappresentato da una DS
  *nome*_name char(255) -> percorso file
  *nome*_nl uns(10) -> lunghezza percorso file
  *nome*_fo uns(10) -> una costante che indica se creare il file, sovrascriverlo o aggiungere contenuto al file già esistente
  queste variabili non possono essere inizializzate
- precompilatore definisce anche le costanti per la variabile *nome*_fo
  - sqfrd → 2
  - sqfcrt → 8
  - sqfovr → 16
  - sqfapp → 32

Scrittura di in un file con CLOB_FILE

```
dcl-s MioFile sqltype(CLOB_FILE);
[...]
MioFile_name = %trim(xml_path) + %trim(xml_name);
MioFile_nl = %len(%trim(MioFile_name));
MioFile_fo = SQFOVR;
```

- Una volta valorizzate queste 3 variabili all'interno del programma RPG, tutto il resto che concerne la gestione del file non è una nostra preoccupazione
- Per scrivere basta eseguire una istruzione SQL **set**

```
exec sql
  set :MioFile = 'Riga di testo da scrivere sul file';
```

Lettura di un file con CLOB_FILE

- Dichiarando sempre una variabile di tipo CLOB_FILE posso leggere un file di IFS e utilizzarne il contenuto per valorizzare una variabile di RPG o un campo di un database

```
dcl-s MioFile sqltype(clob_file);
dcl-s MiaVar varchar(1000);

MioFile_name = '/home/mk1sample/Power coffee 2021-25 a.txt';
MioFile_nl = %len(%trim(MioFile_name));
MioFile_fo = SQFRD;
exec sql
  set :MiaVar = :MioFile;
exec sql
  update MK1SAMPLE/EMP_RESUME
    set RESUME = :MioFile
    where EMPNO = '000190' and RESUME_FORMAT = 'ascii';
```

Tipi dati SQL - CLOB

- **clob:*n*** → in RPG è rappresentato da una DS

  *nome*_data char(*n*) -> contenuto

  *nome*_len uns(10) -> lunghezza del contenuto

  - Lunghezza massima 16.773.100
  - CCSID default 1208
  - non possono essere inizializzate

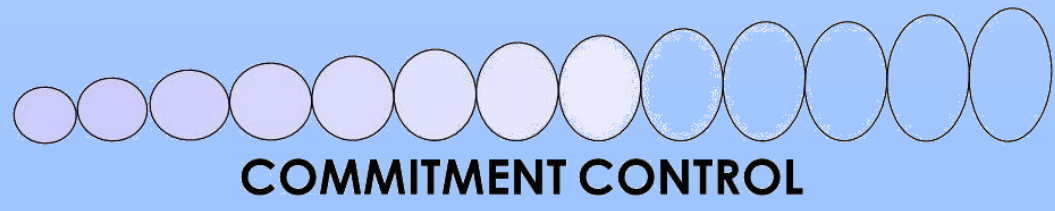
Lettura di un file tramite GET_CLOB_FROM_FILE

- La funzione **GET_CLOB_FROM_FILE** restituisce un CLOB locator con i dati letti da uno stream file o file fisico di origine. P.es.

```
get_clob_from_file('/home/mk1sample/test.txt')
get_clob_from_file('MK1TESTXML/QRPGLESRC(XMLTOIFS1)')
```

- Il risultato della funzione può essere memorizzato in una variabile RPG di tipo dati CLOB

```
dcl-s MioFile sqltype(clob:5000);
Percorso = '/home/mk1sample/Power coffee 2021-25 a.txt';
exec sql
  set :MioFile = get_clob_from_file(:Percorso);
```



Leggere/scrivere con i servizi SQL di IBM i

- Le funzioni di tabella **IFS_READ, IFS_READ_BINARY, IFS_READ_UTF8** leggono uno stream file restituendo il contenuto come carattere, binary o caratteri in formato UTF-8
- Le procedure **IFS_WRITE, IFS_WRITE_BINARY, IFS_WRITE_UTF8** scrivono uno stream file con dati di tipo carattere, binario o carattere in formato UTF-8

Vedi Power coffee 11/2021

da 7.3

Leggere con IFS_READ

- Il contenuto può essere restituito per intero oppure diviso specificando una lunghezza desiderata (default 2 Gb) o il carattere di fine linea (p.es. CR o CRLF, ecc.)
- Il carattere di fine linea specificato nel parametro END_OF_LINE non viene restituito

```
dcl-s Percorso varchar(256);
dcl-s NumeroRiga uns(10);
dcl-s RigaFile varchar(500);

exec sql
  declare csv cursor for
    select LINE_NUMBER, LINE
      from table (
       IFS_READ(PATH_NAME => :PERCORSO, END_OF_LINE => 'ANY')
                  );

Percorso = '/home/mk1sample/Power coffee 2021-25 b.csv';

exec sql
  open csv;

dow *on;
  exec sql
    fetch next
      from csv
      into :NumeroRiga, :RigaFile;
…
```

Leggere un file csv con IFS_READ/1

da 7.3 TR6

- Lavorando in unione con la funzione SQL **SPLIT** o la funzione RPG **%split** è possibile leggere riga per riga un file CSV da IFS splittando i campi in base al carattere delimitatore

```
with
 RIGHE as (
 select LINE_NUMBER, LINE
  from table(IFS_READ(PATH_NAME => :FilePath, END_OF_LINE => 'ANY'))),
 COLONNE as (
 select RIGHE.*, ORDINAL_POSITION as COLONNA, ELEMENT as CAMPO
   from RIGHE cross join table(SYSTOOLS/SPLIT(LINE, ';')) as SL)
 select LINE_NUMBER as "Riga",
        min(case when COLONNA = 1 then CAMPO end) as "Articolo",
        min(case when COLONNA = 2 then CAMPO end) as "Descrizione",
        min(case when COLONNA = 3 then CAMPO end) as "Prezzo"
   from COLONNE
   where LINE_NUMBER > 1 -- per omettere la riga di intestazione del CSV
   group by LINE_NUMBER;
```

cfr. anche il gist di Birgitta Hauser https://gist.github.com/BirgittaHauser/53bfoca980e44daeb764bd8b828be2d2

Power coffee 2021-11 - esempio.csv

```
1 ARTICOLO;DESCRIZIONE;PREZZO
2 ABC;TAVOLO;220,34
3 XYZ;SEDIA;35,00
4 HJK;SGABELLO;
```

Riga	Articolo	Descrizione	Prezzo
2	ABC	TAVOLO	220,34
3	XYZ	SEDIA	35,00
4	HJK	SGABELLO	

Leggere un file csv con IFS_READ/2

da 7.4
TR4
7.3 TR10

da 7.4
TR3
7.3 TR9

- La funzione RPG **%split** può essere utilizzata per valorizzare un array oppure in combinazione con codice operativo **for-each**

```
dcl-s Campi varchar(100) dim(3);
dcl-s Campo varchar(100);
dcl-s i uns(3);
dcl-ds OutPut inz;
  Articolo char(15);
  Descrizione char(50);
  Prezzo packed(13:2);
end-ds;
// divido la riga in sottocampi nell'array Campi
Campi = %split(RigaFile:';');
// elaboro i campi dell'array

// oppure eseguo un ciclo per elaborare ogni singolo sottocampo
//  senza utilizzare l'array
clear i;
for-each Campo in %split(RigaFile:';');
// elaborazione singolo campo
  i += 1;
  select;
    when i = 1;
      Articolo = Campo;
    when i = 2;
      Descrizione = Campo;
    when i = 3;
      Prezzo = %dec(Campo:13:2);
  endsl;
endfor;
```

Scrivere un file con IFS_WRITE

- Il file può essere scritto creando un nuovo file oppure accodando il contenuto ad uno già esistente oppure rimpiazzando un file già esistente
- Limite dimensione file: 2 Gb
- E' possibile eventualmente specificare il carattere di fine linea
- `ifs_write` genera il file nel ccsid corrispondente al job, invece `ifs_write_utf8` genera il file con ccsid 1208

```
exec sql
  call ifs_write(path_name => :Percorso,
                 line => :Testo,
                 overwrite => 'REPLACE',
                 end_of_line => 'CRLF');
```

Riferimenti

- E-mail aziendale: mriva@sirio-is.it
- Blog: www.markonetools.it
- E-mail blog: info@markonetools.it
- Linkedin: www.linkedin.com/in/marcoriva-mk1
- Twitter: @MarcoRiva73
- Facebook: https://www.facebook.com/markonetools/
- YouTube: https://www.youtube.com/channel/UCb47YJQJCzU-5x4nnGzDu-w