IBC233 - System i Business Computing

Lecture 10: RPG Programming with Printer Files

Agenda

- RPG Programming with Printer Files

SQL and RPG

- When defining a table, SQL doesn’t follow the rules that the record format name must be different than the table name.
- The RPG program does not allow the record format to be the same as the file/table name.

Solve the SQL problem – F Spec change

Filename: This is where you type the pf or table name

FT (File Type): I O U C

FD (File Designation): P R S T F
(we use blank or F)

FF (File Format): F E

Record Length: blank or a number

RAT (Record Address Type): A D F K P D Z
(we use blank or K)

DEVICE: Printer, Disk, WorkStn

Solve the SQL problem – F Spec change

- Specifying RENAME option on the RPG F-spec so that the format is defined with a different name internal to the RPG program:

- KEYWORDS:
  RENAME(fileName:newRecName)

- Example:

```
FSHIFTRATESIF   E             DISK    RENAME(SHIFTRATES:SHIFTRATER)
FALLSHIFT  IF   E           K DISK    RENAME(ALLSHIFT:ALLSHIFTR )
```

Defining a Spooled File

Filename: Name of the file

FT (File Type): O

FD (File Designation): blank

FF (File Format): E

DEVICE: Printer

KEYWORDS: OFLIND(*IN01)

- OverFLow INDicator

Example:

```
FPAYRPT    O    E             PRINTER OFLIND(*IN01)
```

RCDFMT in SQL

- RCDFMT in SQL (Creating view)
  - In the native database, files have record formats. It actually allows you to have multiple formats in a single file.
  - SQL does not support this. so when you create a file with SQL, a record format is assumed to be the same name as the file.
  - The RCDFMT allows you to override that behavior.

File Overrides

- OVRDBF
  - A CLLE command: allows you to use a different file instead of the one that is defined by the RPG program
  - Example

    OVRDBF ALLSHIFT NIGHTS

    - Override all reference to the file ALLSHIFT to be directed to the view NIGHTS
- DSPOVR

File Overrides

- OVRPRTF
  - Changes attributes about the spool file

Rounding Numbers in RPG

```
Eval (h)
```

```
Eval (h) hourlyrate =
         hourlyrate * 1.07;
```

Logic for an RPG program that creates a report for all of the records in a file

To print lines on the Report…

- Write *recordname*

Report Logic

Initialize Variables

Report Logic

Initialize Variables

Print the report heading

Report Logic

Initialize Variables

Print the report heading

Read the first record and check for EOF

Report Logic

Initialize Variables

Print the report heading

Read the first record and check for EOF

Loop until EOF

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
```

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
    update the totals
```

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
    update the totals
    print the detail line
```

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
    update the totals
    print the detail line
    read the next record
```

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
    update the totals
    print the detail line
    read the next record
End of loop
```

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
    update the totals
    print the detail line
    read the next record
End of loop
Print the totals
```

Report Logic

```
Initialize Variables
Print the report heading
Read the first record and check for EOF
Loop until EOF
    format the detail line
    update the totals
    print the detail line
    read the next record
End of loop
Print the totals
End of program logic
```