The Power of VIEWs and the New ViewManager

Mike Hanson
President
BoxSoft Development Inc.
September 24, 1997

Overview

- What are VIEWs?
  - How can they help me?
  - How do I use them?
- What is the ViewManager?
  - Where does it fit?
  - How do I use it?
- MHView Template

VIEWs

- What is a VIEW?
  - A "VIEW" is an abstraction of a file or group of files.
  - It enables complex file access, keeping the details hidden.
  - It's helpful, even for one file.
  - It's a huge timesaver for multiple related files.

Declaring VIEWs

- Required
  - Primary File
- Optional
  - Projected Fields
  - Related Files (with projected fields)
  - Filter
  - Order

Examples of VIEWs

```
ViewCustomer    VIEW(Customer).

ViewCusInv      VIEW(Customer)
                  JOIN(Inv:CusKey, Cus:No)
                    PROJECT(Inv:Date)
                . .

ViewInvCus      VIEW(Invoice)
                  JOIN(Cus:NoKey, Inv:CusNo)
                    PROJECT(Cus:Name)
                . .
```

Filters

- Controls which records are retrieved.
- Uses keys in primary file, if possible.
  - Consider changing your primary file to make use of this feature. VIEWs can be constructed starting with any file in the schema.

Examples of Filters

- Examples
  - Within VIEW Structure
    ```
    FILTER('Inv:Date = TODAY()')
    ```
  - Run-time Propertery Assignment
    ```
    View{PROP:Filter} = 'Inv:Date = TODAY()'
    ```
- All variables and functions must be bound.

Orders

- Controls the sequence in which records are retrieved.
- Uses fields from any file in the VIEW.
- Uses keys from the primary file, if possible.
- Non-keyed orders are processed in memory.
- Can be slow, if you're not careful.

Examples of Orders

- Examples
  - Within VIEW Structure
    ```
    ORDER('+Cus:Name,-Inv:Date')
    ```
  - Run-time Propertery Assignment
    ```
    View{PROP:Order} = '+Cus:Name,-Inv:Date'
    ```
- Fields or expressions may be used.
- All fields and functions must be bound.

Turning VIEWs Inside-Out

- Most of the time, views can be started with any file as the primary file.
- Determine which file contains the best keys for the Filter and Order settings, make it the primary file, then reattach the other files.
- Use a Filter to prevent unwanted records. (Be aware of the "forced primary record".)

Inside-Out Example

```
View        VIEW(Customer)
              JOIN(Inv:CusKey, Cus:No)
                PROJECT(Inv:No)
                PROJECT(Inv:Date)
                JOIN(Itm:InvKey, Inv:No)
                  PROJECT(Itm:PrdNo)
            . . .
View        VIEW(Invoice)
              JOIN(Cus:NoKey, Inv:CusNo)
                PROJECT(Cus:Name)
              .
              JOIN(Itm:InvKey, Inv:No)
                PROJECT(Itm:PrdNo)
            . .
```

Saving Time with VIEWs

- Much hand coded file access can be prevented with VIEWs.
  - For example:
    Count all Customers without any Invoices.
    (a.k.a.: Outer Join)

Hand Code

```
SET(Customer)
LOOP
  NEXT(Customer);  IF ERRORCODE() THEN BREAK.
  CLEAR(Inv:Record,-1)                              !**
  Inv:CusNo = Cus:No                                !**
  SET(Inv:CusKey, Inv:CusKey)                       !**
  NEXT(Invoice)
  IF NOT ERRORCODE() AND Inv:CusNo = Cus:No         !**
    Count# += 1
  END
END
```

View Code

```
View        VIEW(Customer), FILTER(Inv:Date=0)
              JOIN(Inv:CusKey, Cus:No)
                PROJECT(Inv:Date)
            . .

  OPEN(View)
  LOOP
    NEXT(View); IF ERRORCODE() THEN BREAK.
    Count# += 1
  END
  CLOSE(View)
```

VIEW Summary

- Views allow you to access one or more related files.
- You can restrict which records are processed with a Filter.
- You can control the sequence with an order.
- You can turn them inside-out.
- You can save yourself time.

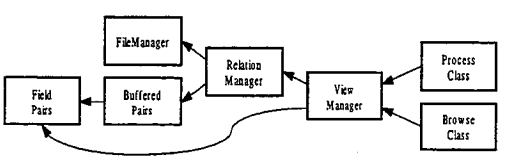
ViewManager Class

Everything you ever wanted!
(and a little bit more)

ViewManager

- What is it?
  - New CLASS in Clarion 4's ABC Templates.
  - Handles details of complex view manipulation.
  - Probably overkill if you are hand-coding (unless a template is writing the code for you).

Where does it fit?



What does it do?

- Handles multiple "Sort Orders"
  - Think of different tabs on a Browse
- Handles Order clause for each Sort Order.
- Handles Filter clause for each Sort Order.
- Validate Records (virtual function applied externally after the regular Filter).
- Primes fields for new records.

How do you use it?

- Instantiate the class in your data area.
- Initialize the object.
- Add one or more Sort Orders
  - Specify additional order fields.
  - Specify one or more prioritized filters (for ranges, miscellaneous filters, etc.)
- Process the records

This Is Hard!!!

- It's probably overkill, if you are hand-coding your view operations.
- The class interface is as awkward as manual file access statements.
- The class's primary purpose is to enable Browse and Process templates to share common code.

How can we use it?

- Hand coding is possible, but difficult.
- The Process procedure template is useful, but sometimes is overkill.
- An extension template can be written to hide the complexity, providing us with a gentle programming interface.

MHView Template

VIEWs for the rest of us.

MHView Template

- Add it to any type of Procedure
  - Source (with mhSourceFiles)
  - Browse
  - Process
  - Window

MHView - Template Settings

- File schematic (in the Files window).
- Name (in case you have more than one)
- Filter
- Range Limits
- Additional Field Pairs
- Order

MHView - Code Algorithm

- Call the "Open" routine
- Process the View
- Call the "Close" routine

MHView - Sample Code

```
DO OpenView
LOOP
   NEXT(View); IF ERRORCODE() THEN BREAK.
   !DO Something
END
DO CloseView
```

Where can you get it?

- It's FREE!!!
- Surf to www.BoxsoftDevelopment.com
- Go to the Download area
- Look for MHTPL*.ZIP

Conclusion

- VIEWs
  - The VIEW structure enables you to perform complex file access operations with minimal code.
  - It will exclude unwanted records, and sort the results into any desired sequence.

Conclusion

- ViewManager Class
  - Extends the power of the Views by adding multiple sort orders, stacked filters, and field priming.
- MHView Template
  - Enables programmers to access the power of VIEWs and the ViewManager in their own code.