Table of Contents

[Screen Shot of EXTVIEW02 2](#_Toc323899526)

[Overview 2](#_Toc323899527)

[Forms Startup 2](#_Toc323899528)

[Main Menu 3](#_Toc323899529)

[Exit Button 3](#_Toc323899530)

[Data Fields (#1) 3](#_Toc323899531)

[Record processing 3](#_Toc323899532)

[Buttons (#2) 3](#_Toc323899533)

[View Journal 3](#_Toc323899534)

[The Data Fields 4](#_Toc323899535)

[Newer/Older Buttons 4](#_Toc323899536)

[Close Window Button 4](#_Toc323899537)

[Comment 4](#_Toc323899538)

[TRIGGER CODE 5](#_Toc323899539)

[PRE-INSERT 5](#_Toc323899540)

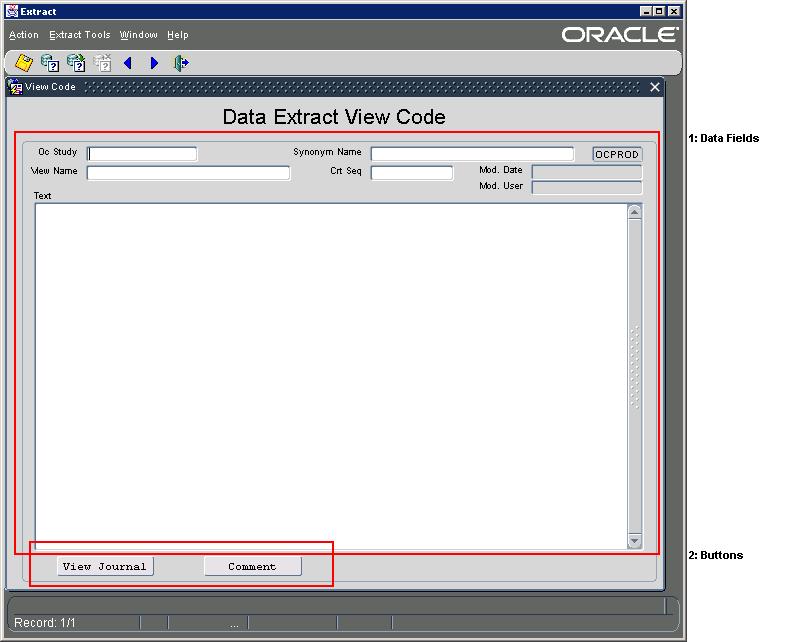
[PRE-UPDATE 5](#_Toc323899541)

[PRE-DELETE 6](#_Toc323899542)

Detailed analysis of EXTVIEW02.FMB

Note: Used TEXTPAD to look inside form as the 6i Form Builder was unable to read form.

# Screen Shot of EXTVIEW02



# Overview

This form is used to manipulate the cursor SQL code for a File Id of a defined extraction. It also allows the user to see previous version of the same cursor.

# Forms Startup

1. Security Checked at form startup. The following query checks userid to see if user is valid for APPLICATION. (WHEN-NEW-FORM-INSTANCE)

SELECT '1' INTO T\_WHAT

FROM CT\_EXT\_ACCOUNTS

WHERE (USER\_NAME = USER or User = 'CTEXT\_T')

AND ROWNUM = 1;

1. Menu Actions are turn off that would normally allow queries and editing.

# Main Menu

The main menu controls what other screens can be selected. They are controlled through ROLES. Only certain Oracle ROLES granted to the user would allow access. See EXTMENU1\_Analysis for more information.

# Exit Button

The exit button is used to close the application. When closing the application, the screen is checked for existing non-committed transactions and will not quite until the transactions are committed or rolled back.

# Data Fields (#1)

There are 7 enterable fields that make up the data field for a cursor definition. They are stored in the table CT\_EXT\_VW\_CTL.

Functions required: Enter/execute Query, EDIT/UPDATE, DELETE (May have to look into ROLE based display of these options)

Note: When a record is modified, the “before modified” version of the record must be copied to the Cursor Journal table. Also, a Comment must be captured when a record is modified.

|  |  |  |
| --- | --- | --- |
| Screen Field | Table Column | Description |
| OC Study | OC\_STUDY | Study identifier |
| View Name | VIEW\_NAME | The name of the view object. |
| Synonym Name | SYNONYM\_NAME | The Synonym name of the view object |
| Crt Seq | CRT\_SEQ | The sequence number order in which the View should be created. |
| Text | TEXT | This is the actual SQL code that makes up the select statement that will build the view. NOTE: This is stored in a LONG column. |
| Mod. Date | MODIFIED\_BY | User who last modified the record. |
| Mod. User | MODIFIED\_DATE | The date the records was modified. |
|  | CREATE\_BY | User who created record, NOT DISPLAYED |
|  | CREATION\_DATE, | Date record created, NOT DISPLAYED |
|  | NOTE | Comment field used to explain/describe modifications NOT DISPLAYED. |

### Record processing

DELETES: Before a record is committed as a delete from the table, a copy of it is placed into the Journal Table, and journaling information is stored with the record. See PRE-DELETE trigger for entire PL/SQL code.

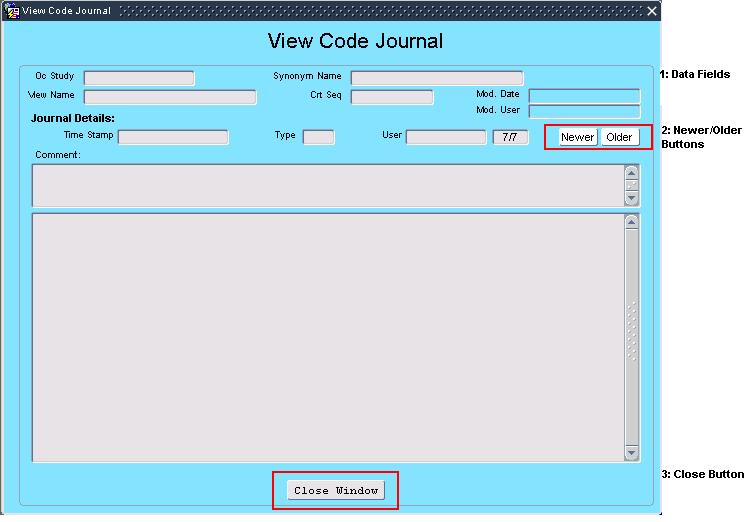
UPDATES: Before a record is committed as an update, a copy of it is placed into the Journal Table, and journaling information is stored with the record. See PRE-UPDATE trigger for entire PL/SQL code.

INSERTS: Before a new record is committed, a copy of it is placed into the Journal Table, and journaling information is stored with the record. See PRE-UPDATE trigger for entire PL/SQL code.

# Buttons (#2)

### View Journal

The View Journal Button is used to query past cursor definitions versions.



* Only fires if OC\_STUDY and VIEW\_NAME fields have been entered.
* Displays all journal entries for the cursor displayed, using the following query:

where CT\_EXT\_VW\_CTL$JN.OC\_STUDY = :CT\_EXT\_VW\_CTL.OC\_STUDY

and CT\_EXT\_VW\_CTL$JN.VIEW\_NAME = :CT\_EXT\_VW\_CTL.VIEW\_NAME

#### The Data Fields

The data fields are read-only and come from the table CT\_EXT\_VW\_CTL$JN. Data is sorted by most recent record.

#### Newer/Older Buttons

These buttons will traverse the user through the journal records.

#### Close Window Button

This button is used to close the journal window and return the user to the primary data entry form.

### Comment

The comment button is used to place a comment on the current record. Comments are also collected during updates and deletes.

# TRIGGER CODE

#### PRE-INSERT

DECLARE

T\_TEXT VARCHAR2(120) := ' ';

BEGIN

T\_TEXT := ' Inserting journaling information failed.';

Insert into CT\_EXT\_VW\_CTL$JN (

JN\_OPERATION ,JN\_TIMESTAMP

,JN\_SN

,JN\_ORACLE\_USER

,OC\_STUDY ,VIEW\_NAME

,CRT\_SEQ ,TEXT

,SYNONYM\_NAME ,CREATE\_BY

,CREATION\_DATE ,NOTE )

VALUES (

'INS' ,SYSDATE

,TO\_NUMBER(TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS'))

,USER

,:CT\_EXT\_VW\_CTL.OC\_STUDY ,:CT\_EXT\_VW\_CTL.VIEW\_NAME

,:CT\_EXT\_VW\_CTL.CRT\_SEQ ,:CT\_EXT\_VW\_CTL.TEXT

,:CT\_EXT\_VW\_CTL.SYNONYM\_NAME ,USER

,SYSDATE ,:CT\_EXT\_VW\_CTL.NOTE

);

IF NVL(:CT\_EXT\_VW\_CTL.NOTE, ' ') <> NVL(:CT\_EXT\_VW\_CTL.NOTE\_DISPLAY, ' ') THEN

:CT\_EXT\_VW\_CTL.NOTE\_DISPLAY := :CT\_EXT\_VW\_CTL.NOTE;

END IF;

EXCEPTION

WHEN OTHERS THEN

MESSAGE(T\_TEXT, ACKNOWLEDGE);

RAISE FORM\_TRIGGER\_FAILURE;

END;

#### PRE-UPDATE

DECLARE

T\_TEXT VARCHAR2(120) := ' ';

H\_OC\_STUDY CT\_EXT\_VW\_CTL.OC\_STUDY%Type;

H\_VIEW\_NAME CT\_EXT\_VW\_CTL.View\_Name%Type;

H\_CRT\_SEQ CT\_EXT\_VW\_CTL.CRT\_SEQ%Type;

H\_SYNONYM\_NAME CT\_EXT\_VW\_CTL.Synonym\_Name%Type;

H\_CREATE\_BY CT\_EXT\_VW\_CTL.Create\_By%Type;

H\_CREATION\_DATE CT\_EXT\_VW\_CTL.Creation\_Date%Type;

H\_TEXT CT\_EXT\_VW\_CTL.Text%Type;

H\_NOTE CT\_EXT\_VW\_CTL.Note%Type;

H\_Mod\_by CT\_EXT\_VW\_CTL.Modified\_by%Type;

H\_Mod\_date CT\_EXT\_VW\_CTL.Modified\_date%Type;

h\_rowid varchar2(30) :=NAME\_IN(:SYSTEM.CURRENT\_BLOCK || '.ROWID');

BEGIN

T\_TEXT := ' Updating journaling information failed.';

:ct\_ext\_vw\_ctl.modified\_by := user;

:ct\_ext\_vw\_ctl.modified\_date := sysdate;

select OC\_STUDY, VIEW\_NAME, CRT\_SEQ, TEXT,

SYNONYM\_NAME, CREATE\_BY, CREATION\_DATE, NOTE,

MODIFIED\_BY, MODIFIED\_DATE

into H\_OC\_STUDY, H\_VIEW\_NAME, H\_CRT\_SEQ, H\_TEXT,

H\_SYNONYM\_NAME, H\_CREATE\_BY, H\_CREATION\_DATE, H\_NOTE,

H\_MOD\_BY, H\_MOD\_DATE

from CT\_EXT\_VW\_CTL

where rowid = h\_rowid;

Insert into CT\_EXT\_VW\_CTL$JN (

JN\_OPERATION ,JN\_TIMESTAMP

,JN\_SN

,JN\_ORACLE\_USER

,OC\_STUDY ,VIEW\_NAME

,CRT\_SEQ ,TEXT

,SYNONYM\_NAME ,CREATE\_BY

,CREATION\_DATE ,NOTE

,MODIFIED\_BY ,MODIFIED\_DATE )

VALUES (

'UPD' ,SYSDATE

,TO\_NUMBER(TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS'))

,USER

,H\_OC\_STUDY ,H\_VIEW\_NAME

,H\_CRT\_SEQ ,H\_TEXT

,H\_SYNONYM\_NAME ,H\_CREATE\_BY

,H\_CREATION\_DATE ,H\_NOTE

,H\_MOD\_BY ,H\_MOD\_DATE);

IF NVL(:CT\_EXT\_VW\_CTL.NOTE, ' ') <> NVL(:CT\_EXT\_VW\_CTL.NOTE\_DISPLAY, ' ') THEN

:CT\_EXT\_VW\_CTL.NOTE\_DISPLAY := :CT\_EXT\_VW\_CTL.NOTE;

END IF;

EXCEPTION

WHEN OTHERS THEN

MESSAGE(T\_TEXT, ACKNOWLEDGE);

RAISE FORM\_TRIGGER\_FAILURE;

END;

#### PRE-DELETE

DECLARE

T\_TEXT VARCHAR2(120) := ' ';

H\_OC\_STUDY CT\_EXT\_VW\_CTL.OC\_STUDY%Type;

H\_VIEW\_NAME CT\_EXT\_VW\_CTL.View\_Name%Type;

H\_CRT\_SEQ CT\_EXT\_VW\_CTL.Crt\_Seq%Type;

H\_SYNONYM\_NAME CT\_EXT\_VW\_CTL.Synonym\_Name%Type;

H\_CREATE\_BY CT\_EXT\_VW\_CTL.Create\_By%Type;

H\_CREATION\_DATE CT\_EXT\_VW\_CTL.Creation\_Date%Type;

H\_TEXT CT\_EXT\_VW\_CTL.Text%Type;

H\_NOTE CT\_EXT\_VW\_CTL.Note%Type;

H\_Mod\_by CT\_EXT\_VW\_CTL.Modified\_by%Type;

H\_Mod\_date CT\_EXT\_VW\_CTL.Modified\_date%Type;

h\_rowid varchar2(30) :=NAME\_IN(:SYSTEM.CURRENT\_BLOCK || '.ROWID');

BEGIN

T\_TEXT := ' Deleting journaling information failed.';

select OC\_STUDY, VIEW\_NAME, CRT\_SEQ, TEXT,

SYNONYM\_NAME, CREATE\_BY, CREATION\_DATE, NOTE,

MODIFIED\_BY, MODIFIED\_DATE

into H\_OC\_STUDY, H\_VIEW\_NAME, H\_CRT\_SEQ, H\_TEXT,

H\_SYNONYM\_NAME, H\_CREATE\_BY, H\_CREATION\_DATE, H\_NOTE,

H\_MOD\_BY, H\_MOD\_DATE

from CT\_EXT\_VW\_CTL

where rowid = h\_rowid;

Insert into CT\_EXT\_VW\_CTL$JN (

JN\_OPERATION ,JN\_TIMESTAMP

,JN\_SN

,JN\_ORACLE\_USER

,OC\_STUDY ,VIEW\_NAME

,CRT\_SEQ ,TEXT

,SYNONYM\_NAME ,CREATE\_BY

,CREATION\_DATE ,NOTE

,MODIFIED\_BY ,MODIFIED\_DATE )

VALUES (

'DEL' ,SYSDATE

,TO\_NUMBER(TO\_CHAR(SYSDATE, 'YYYYMMDDHH24MISS'))

,USER

,H\_OC\_STUDY ,H\_VIEW\_NAME

,H\_CRT\_SEQ ,H\_TEXT

,H\_SYNONYM\_NAME ,H\_CREATE\_BY

,H\_CREATION\_DATE ,H\_NOTE

,H\_MOD\_BY ,H\_MOD\_DATE);

IF NVL(:CT\_EXT\_VW\_CTL.NOTE, ' ') <> NVL(:CT\_EXT\_VW\_CTL.NOTE\_DISPLAY, ' ') THEN

:CT\_EXT\_VW\_CTL.NOTE\_DISPLAY := :CT\_EXT\_VW\_CTL.NOTE;

END IF;

EXCEPTION

WHEN OTHERS THEN

MESSAGE(T\_TEXT, ACKNOWLEDGE);

RAISE FORM\_TRIGGER\_FAILURE;

END;