**OMNI**

It is a record keeping system. Used to store records for retirement services.

It mainly uses 7 VSAM master files:

1. PLAN

2. PARTICIPANT (PART)

3. FUND

4. TRANSACTION (TRAN)

5. UCOM

6. HISTORY

7. ARCHIVED HISTORY

**1. PLAN**

* The PLAN master file consists of set of rules for any retirement policies.
* It is a 6-digit number.
* To browse plans we can use PLPLBR
* To create a plan object we can use PLPLOBJ
* T613 transaction is used for plan maintainence.
* PLLCBR can be used to search to which segment a plan belongs.

**2. PARTICIPANT (PART)**

* Participant is aperson who is getting enrolled to a plan either by himself or by his company.
* This master file consists of all the participant details. For example participant name, social security number, contact information etc.
* It is of total 13 bytes which consists of 9 bytes social security number(SSN) and additional 4 bytes depending on the participant.
* To browse participant we can use PTPHBR
* To create a participant object we can use PTPHOBJ
* T813 transaction is used for participant maintainence.

**3. FUND**

* This file consists of number of funds under each plan.
* Contribution:

It is the money put towards the plan.

Contributions that are made by employees towards their plan is known as employee contribution.

Contributions that are made by employers towards their employee plan is known as employee contribution.

Defined Contribution -

Employees contribute a fixed amount or a percentage of their paycheck to an account that is intended to fund their retirement.

Some times the sponsor company will match a portion of employee contributions as an added benefit.

Defined Benefit -

Employer sponsored retirement plan where employee benefits are composed using a formula that considers several factors such as length of employment and salary history.

Pre-tax contribution -

The contribution is made before any taxes are paid on the amount.

Advantage is that they can reduce your taxable income.

Post-tax contribution -

The contribution is made after paying the taxes on the amount.

* It is a 3-byte field where 2 bytes are for FUND ID which is the combination of from where is the fund coming and where is it going and 1 byte FUND SOURCE which tells the source of the fund.
* The main fund sources are:

A n D, where A stands for employee and D stands for employer.

* PTPFINQ can be used for fund inquiry.
* There are many transactions that can be used to take loans or withdraw amount(disbursement) - T381, T383, T384, T385.
* There can be two types of withdrawal:

Hardship - If some problem occurs. For example medical emergency.

In-Service - When some age limit has reached.

**4. TRANSACTION (TRAN)**

* Everything is based on transaction.
* Transactions can be created through cards.
* T588(schedule) is a universal transaction.
* Transaction 104 is used for plan level contribution.
* Transaction 114 is used for participant level contributions.

**5. UCOM**

* It is a spool for transactions.
* It consists of report for card getting created, we can see what a card looks like, failure report, success report.

**6. HISTORY**

* It consists of all history related to plan, participant, transactions etc.
* To browse this file we can use HIBRBR.

**7. ARCHIVED HISTORY**

* Histories older than 30 days will get archived.
* AHBRBR can be used for browsing this file.

JOBCALC is used for submitting jobs.

JOBUNIF is used for running folders.

JOBUNIC is used for creating cards.

VTFHBR can be used for folder browsing.

Inside folders /a indicates posted but not permanent.

/e indicates error out in production.

TXTXBR can be used for text browsing.

There are mainly 2 types of DATE used-

TRADE DATE - date on which transaction was added or posted.

RUN DATE - date on which transaction was run into production.

**CONTAINERS:**

Containers are similar to tabels in COBOL.

Functions that deal with tabels start with OCData for example, OCData\_CoCreate , OCData\_CoDestroy,etc.

**DeNum Concept:**

0-499 ->Numeric feilds(represented by nums)

500-599 -> 10 bytes text feild(represented by tx10s)

600-699 -> 40 bytes text feild(represented by tx40s)

700-799 -> 200 bytes text feild(represented by tx200s)

**Container Functions:**

**Creating the container:**

**OCData\_CoCreate(name '**container1**' keylength '**18'**num:1 tx10s:10)**

The above function creates the container with the name container1 and the key of length 18 bytes with 1 feild of type numeric and second feild of type 10 bytes text feild.

**Reading the records of the container:**

**Sequential reading:**

**OCData\_ItemVeiw(Name '** name\_of\_the\_container**')**

**loop while**

**OCData\_ItemNext(**name\_of\_the\_container)

**endloop**

The **OCData\_ItemVeiw()**  reads the content of the container from the first record.

The **OCData\_ItemNext()**  reads next record until there are no records left.

**Random Read:**

**OCData\_ItemVeiw(Name '** name\_of\_the\_container**' key '**key\_feild' **)**

The **OCData\_ItemVeiw()** reads the record of the conatiner with the key specified in the function .

When the complete key is not known the partial key can be passed as a parameter to fetch the records,the key prefix should be spcifed in the function.

**OCData\_ItemVeiw(Name '** name\_of\_the\_container**' keyprefix '**partial\_key\_feild' **)**

**Updating the container value:**

**1)Fetching the value of the row that needs to be updated :**

**OCData\_ItemGet(name ' ' key ' ' )**

**OCData\_ItemInit(name ' ' key ' ' )**

One of the two functions can be used to fetch the record .Using **OCData\_ItemInit()** will overwrite.

**2)Set the value :**

**OCData\_ItemSet(name:' ' DeNum' ' Value '')**

The value of the feild in the record is set using theabove function where name indicates the container name,Denum refers to the feild of the record being fecthed and the value refers to the new value.

**3)Update the value**:

**OCData\_ItemUpdate(name ' ')**

The value that is set using the OCData\_ItemSet is Updated in the container using the OCData\_ItemUpdate() Function.Multiple Feilds in the record can be updated by using multiple **OCData\_ItemSet()** fuction.

**Deleting the container:**

**OCData\_CoDestroy(name '**container\_name**')**

This distroys the container with name specified within the function.

**OCData\_CoDestroyAll()**

Distroys all the container present.

**Files in omni**

**1.File**

No. of bytes per record. File in omni can contain upto 200 characters.

**2.Filv**

If a file contain more than 200 character it is called filv.

Maximum no. of character we can write in filv is 1200.

**NAMING**

We can name a file as:

File1, File2

We can name a filv as

Filv1, Filv2

**Functions which we can perform on file are**

1.OPEN

2.Close

3.Read

4.Write

OCFILE1\_OPEN(NAME : “FILE1” MODE : “INPUT”)

OCFILE1\_READ()

OCFILE1\_WRITE(TX101)

OCFILE1\_CLOSE();

**Data type (pre-defined variables) :**

The following are the pre-defined variables,

\* TX : Alpha Numeric. There will be 1-120 characters.

\* SD : System Defined Variables.

-> It can be named as SD001,SD002......

-> SD003 is for Date, SD008 is to count the loop.

\* WK : Numeric data type.

\* WA : Accumulation

\* KV1 : It is Key Value data type. It can be used for lower date ie., "from"

\* KV2 : It is Key Value data type. It can be used for higher date ie., "to"

**LOOPING :**

**Example :**

*x.udplan = " ";*

*x.udpart = " ";*

*PLPLobj\_view();*

*loop while*

*PLPLobj\_next();*

*udplan = PLPLobj\_De(001)*

*PTPHobj\_view();*

*loop while*

*PTPHobj\_next();*

*udpart = PTPHobj\_De(007);*

*oc\_show("udplan : " udplan "udpart : " udpart)*

*End loop;*

*End loop;*

Here,

\* *PLPLobj\_view()* : It is used for sequential view.

\* *PLPLobj\_next()*  : It is used to read next element.

\* *oc\_show* : It is used to print .