

## **CONTENTS**

|                                       |    |
|---------------------------------------|----|
| INTRODUCTION TO AXPERT .....          | 2  |
| SESSION 1-SQL STATEMENTS.....         | 5  |
| SESSION 2-T-STRUCT CREATION.....      | 8  |
| SESSION 3-MASTER DETAIL& I-VIEWS..... | 22 |
| SESSION 4-T-GEN MAPS.....             | 25 |
| SESSION 5-FILL GRID.....              | 29 |
| SESSION 6-IVIEW PARAMETERS.....       | 35 |
| SESSION 7-SUB FORMS (Popup).....      | 42 |
| SESSION 8- Actions .....              | 43 |

## INTRODUCTION

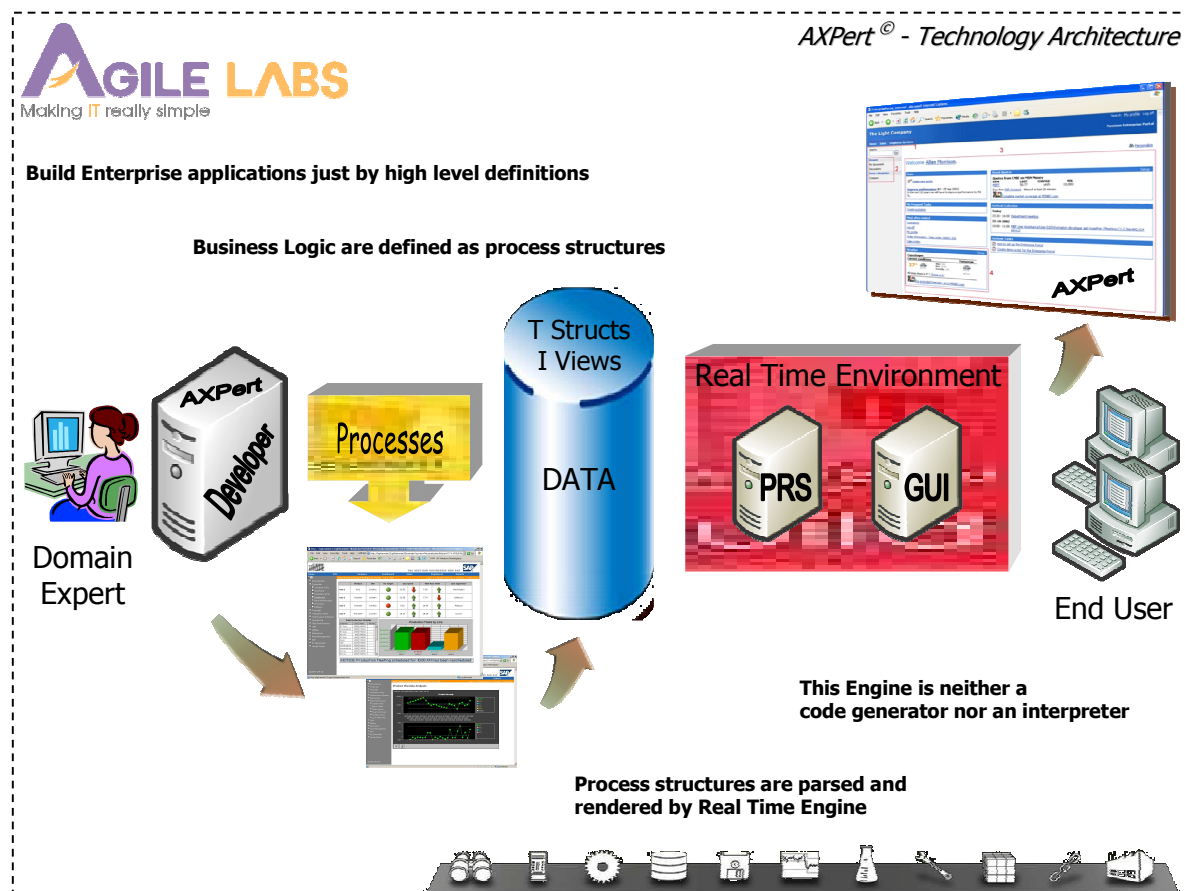
**Axpert** is a tool that can be used to do Option Driven Software Development.

To develop a software application using Axpert, identify the menu options to be provided in the software. Define the menu options using the Axpert developer application. Menu options in any enterprise software may be input forms, queries, reports & utilities.

Input forms are defined as **Transaction Structures (TStructs)** through the Axpert Developer application. Queries & reports are defined as **Information Views (IViews)**.

TStructs and IViews are called as Process Structures. Hence, an application is developed in Axpert by representing the application logic as process structures. The process structures are stored in the database. These are read by the Axpert RTE and rendered at run time. There is no code that is generated to create a software application from the process structures. Axpert works like a browser that parses an HTML file & renders a screen. Similarly, Axpert parses the process structures and renders an application.

## AXPERT ARCHITECTURE



Business process definitions are stored as Transaction structures and Information Views.

Transaction structures are data structures that govern the data input, validation and the storage functionality of an information system. The Axpert RTE displays a transaction form for every transaction structure.

A TStruct is a collection of

- ▲ Data containers
- ▲ GUI Definitions
- ▲ Process maps
- ▲ Print formats

Data Containers are logical groups of input data in a transaction structure. Containers can be classified into simple and tabular. A simple container will hold input fields which will hold only one value in a single transaction. A tabular container will hold input fields in the form of a grid and hence can hold many values in a single transaction. A transaction structure can have any number of containers. Input fields are elements, which hold a piece of data of a transaction. Each container can have one or more input fields. GUI Definitions can be made to set the look and feel of the transaction forms.

Process maps provide a way of relating transaction structures and passing data from one transaction to another. Three kinds of process maps are provided

- ▲ Transaction Generator
- ▲ Master-Detail

**Transaction Generator** maps a source transaction structure to a target transaction structure. Such a mapping will create one or more target transaction automatically when the source transaction is stored.

**Master-Detail** maps are made to update data from a detail transaction into another related transaction called the master transaction. The field in the details is mapped to a field in the master along with the type of update to be done.

Print formats are text documents with input fields embedded in it. A transaction form can be printed along with the data entered in it. The format of the printed document will be the same as the print format with the input fields replaced with actual data. Information views are definitions based on which a query or report is generated. Every Iview generates an Information form.

Information views are of two types :

- ▲ Tabular
- ▲ Free Form

**Tabular View** facilitates definition of sql statements & relating them. It provides provisions to add columns to the result set, attach column expressions & define control breaks.

**Free Form View** consists of a set of SQL statements and a text document with print fields embedded in the document.

Print fields are replaced with actual data before printing the report. The actual data are drawn from the SQL result sets.

## SESSION I

## SQL STATEMENTS

This section introduces you to basic SQL select statements that are required to implement solutions in Axpert. Consider the following 2 tables. We will build SQL to extract data from these 2 tables in various formats.

Table T1

| T1id | Tdate      | TNo |
|------|------------|-----|
| 1    | 01/04/2010 | n1  |
| 2    | 01/04/2010 | n2  |
| 3    | 02/04/2010 | n3  |

Table T2

| T2id | T1id | Name | Amount |
|------|------|------|--------|
| 1    | 1    | A    | 50     |
| 2    | 1    | B    | 75     |
| 3    | 1    | C    | 100    |
| 4    | 2    | D    | 125    |
| 5    | 2    | A    | 150    |
| 6    | 3    | C    | 175    |

1.

Select Tdate, TNo, Name, Amount from T1, T2 where t1.t1id = t2.t1id

Result

| TDate      | TNo | TName | Amount |
|------------|-----|-------|--------|
| 01/04/2010 | n1  | A     | 50     |
| 01/04/2010 | n1  | B     | 75     |
| 01/04/2010 | n1  | C     | 100    |
| 01/04/2010 | n2  | D     | 125    |
| 01/04/2010 | n2  | A     | 150    |
| 02/04/2010 | n3  | C     | 175    |

2.

Select TNo, Sum(Amount) as Amt from T1, T2 where t1.t1id = t2.t1id group by tno

Result

| TNo | Amt |
|-----|-----|
| n1  | 225 |
| n2  | 275 |
| n3  | 175 |

3.

Select Tdate, TNo, Sum(Amount) as amt from T1, T2 where t1.t1id = t2.t1id where amount >= 100 group by tdate, tno

Result

| TDate      | TNo | Amt |
|------------|-----|-----|
| 01/04/2010 | n1  | 100 |
| 01/04/2010 | n2  | 275 |
| 02/04/2010 | n3  | 175 |

❖ Note the result for n1. The records with amount < 100 have been ignored and the remaining have been totaled.

4.

Select Tdate, TNo, Sum(Amount) from T1, T2 where t1.t1id = t2.t1id group by tdate, tno having sum(amount) > 250

Result

| TDate      | TNo | Amt |
|------------|-----|-----|
| 01/04/2010 | n2  | 275 |

❖ Note that the condition in the having clause is applied on the total amount for each tno and not on every record.

5.

Select Name, Sum(Amount) as Amt from t2 group by name

Result

| Name | Amt |
|------|-----|
| A    | 200 |
| B    | 75  |
| C    | 275 |
| D    | 125 |

6.

Consider another table named Names. This will table will be as follows

Names

| NamesId | Name |
|---------|------|
| 11      | A    |
| 21      | B    |

|    |   |
|----|---|
| 32 | C |
| 43 | D |
| 51 | E |

Now in the T2 table, instead of storing the actual names, let it contain a reference to the names table. That is the data in the T2 table will look as follows

The SQL statements 1 should be changed as given below to get the same result after the introduction of this table

Select Tdate, TNo, n.NamesId, Amount from T1, T2, Names n where t1.t1id = t2.t1id and t2.Tname = n.Name

Table T2

| T2id | T1id | NamesId | Amount |
|------|------|---------|--------|
| 1    | 1    | 11      | 50     |
| 2    | 1    | 21      | 75     |
| 3    | 1    | 32      | 100    |
| 4    | 2    | 43      | 125    |
| 5    | 2    | 11      | 150    |
| 6    | 3    | 32      | 175    |

The SQL statement 5 will change to

Select n.Name, Sum(Amount) as Amt from t2, names n where n.namesid = t2.name  
group by name

This session will start you on creating a simple application in Axpert. The application that is to be created is for handling a very simple trading organization that buys items from various suppliers & sells them to various customers.

The proposed software application will have the following input forms

- ▲ Customer Master
- ▲ Supplier Master
- ▲ Product Master
- ▲ Purchase bill
- ▲ Invoice
- ▲ Payments
- ▲ Receipts

Each of the above input forms should be defined as a Transaction Structure in Axpert. The transaction structures are to be defined using the Axpert developer application. Follow the steps given below to create a TStruct.

- ▲ Click on the relevant button in the menu.
- ▲ Set the Transid, Caption properties.
- ▲ Create data container (DC) & set the name and table name properties.
- ▲ Create fields & set the name, caption, mode of entry properties.
- ▲ Save & see in RTE.

Define each of the TStructs as given in the tables below.

**The Supplier Master TStruct**

| Type       | Name         | Caption          | Properties  |
|------------|--------------|------------------|---|
| Tstruct    | Supp         | Supplier Master  |   |
| DC         |              | Supplier Details | TableName = Suppliers   |
| InputField | SupplierName | Supplier Name    | DataType = Characters,<br>DataWidth=150,<br>AllowDuplicate=false,<br>AllowEmpty = false                                   |
| InputField | Address      | Address          | DataType = Characters,<br>DataWidth=300,<br>Form-Component-Memo(This property will accept address in more than one line.) |
| InputField | Taxno        | Tax no.          | DataType = Characters,  |

| Type | Name | Caption | Properties  |
|------|------|---------|---|
|      |      |         | DataWidth = 10,<br>AllowDuplicate = false,<br>AllowEmpty = true |

Save the Tstruct and close the option. Run the Axpert RTE and check.

The system would have created a menu option under the Training folder named 'Supplier Master'. On selecting this option, the system will display an input form in which user can enter data.

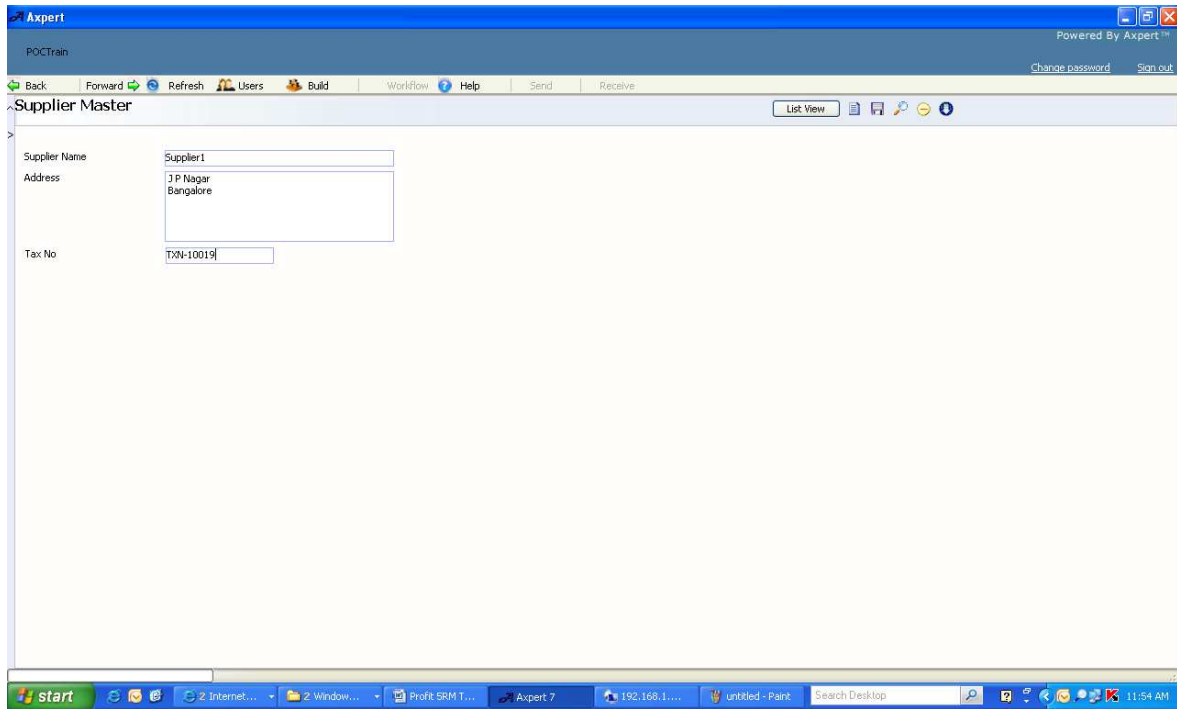
A table named **suppliers** will be created in the database. The fields defined in the T Struct will be created as fields in this table. Additionally a field named **suppliersid** will be created in which a unique number will be generated & stored. This will be used to relate to other tables in the database where ever a relationship with item master has to be established. This field is set as the primary key for the table.

Axpert® will by default provide options to

- ▲ Accept data from end user in the transaction form related to supplier master.
- ▲ Save data to the suppliers table.
- ▲ Load any data for modifications.
- ▲ Store the modifications to the table.
- ▲ Delete a supplier from the table.
- ▲ Validates for duplicate supplier names.

Options to search can be added by adding a search button and selecting the field on which the search should be provided. In this case, the field to search could be set as SupplierName.

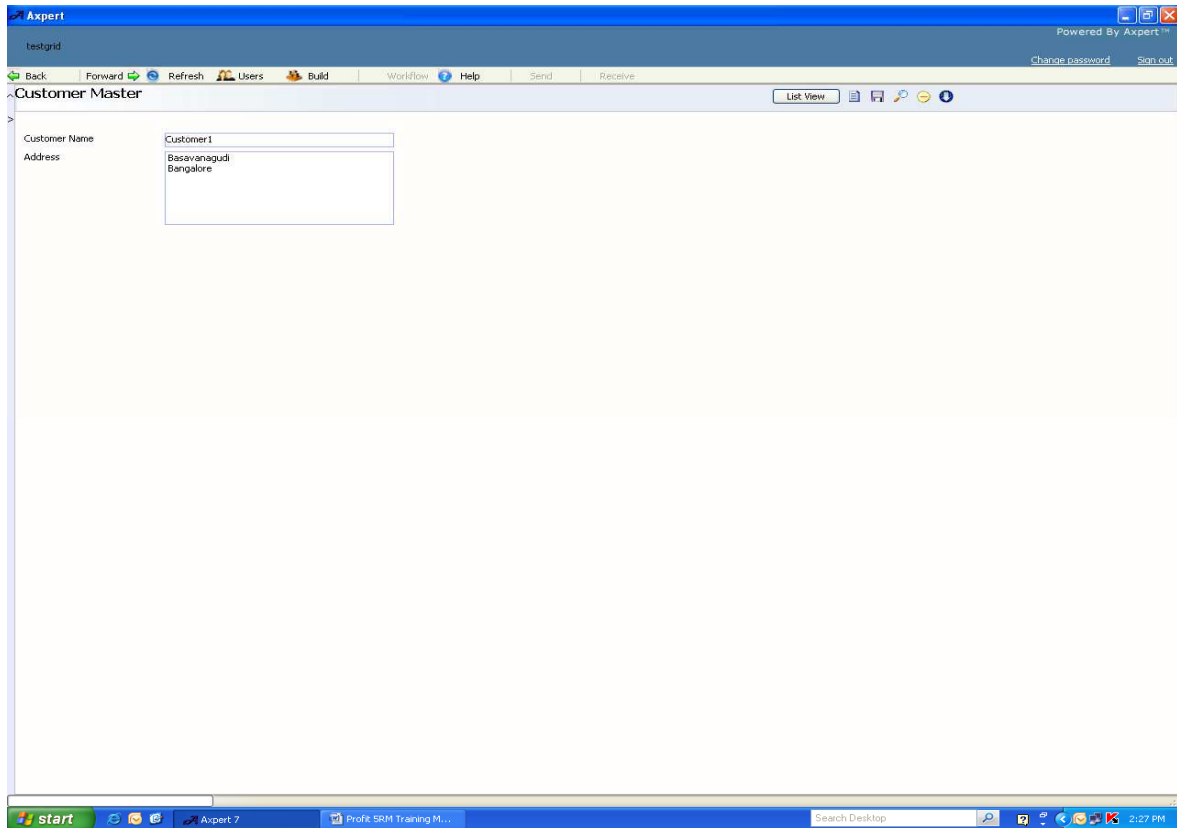




### The Customer Master TStruct

| Type       | Name         | Caption         | Properties  |
|------------|--------------|-----------------|---|
| Tstruct    | Cust         | Customer Master |   |
| DC         |              | Details         | TableName = Customers   |
| InputField | CustomerName | Customer Name   | DataType = Characters,<br>DataWidth=150,<br>AllowDuplicate=false,<br>AllowEmpty = false                                     |
| InputField | Address      | Address         | DataType = Characters,<br>DataWidth = 300,<br>Form-Component-Memo(This property will accept address in more than one line.) |

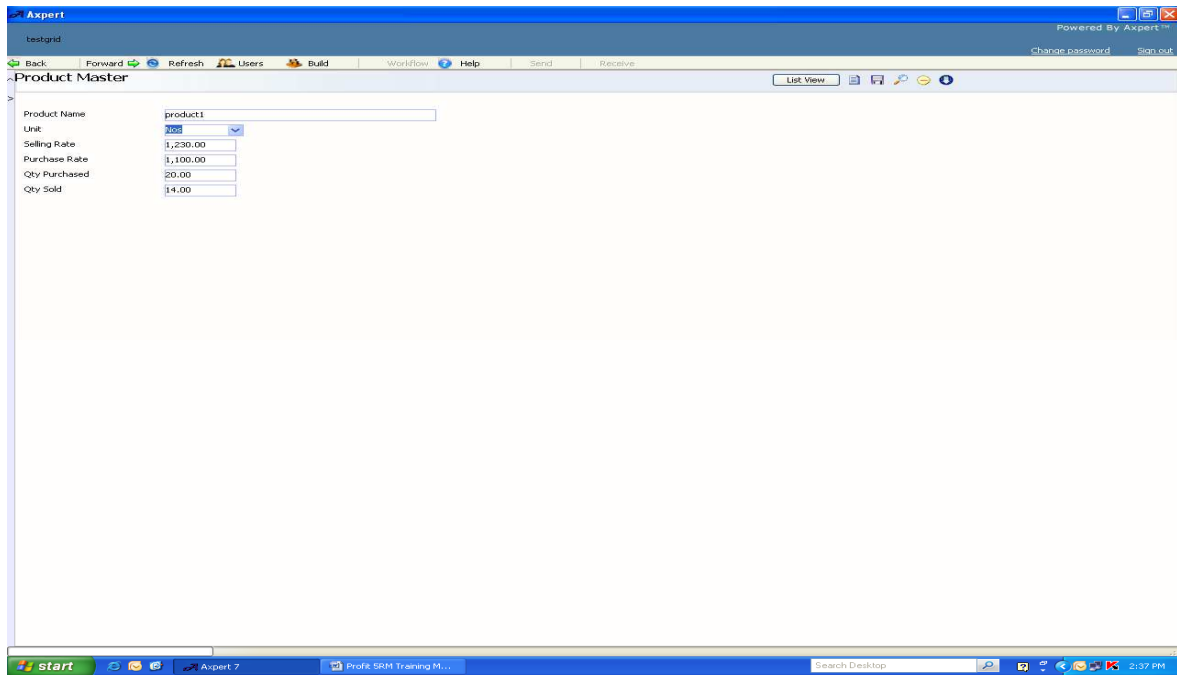
Similar to the supplier form, an input form can be seen in the RTE now. The customers table also will be created in the database. Refer to the discussion in the supplier master Tstruct.



### The Product Master TStruct

| Type       | Name        | Caption        | Properties   |
|------------|-------------|----------------|--|
| Tstruct    | Prod        | Product Master |  |
| DC         |             | Details        | TableName = Products   |
| InputField | ProductName | Product Name   | DataType = Characters ,<br>DataWidth=150,<br>AllowDuplicate=false,<br>AllowEmpty = false     |
| InputField | Unit        | Unit           | DataType= Characters ,<br>Mode of Entry = From List,<br>Details: List Values [Nos, Kgs, Lts] |
| InputField | SellingRate | Selling Rate   | DataType = Numeric(10,2)   |
|            |             |                |  |

| Type       | Name         | Caption       | Properties               |
|------------|--------------|---------------|--------------------------|
| InputField | PurchaseRate | Purchase Rate | DataType = Numeric(10,2) |



### The Purchase Bill TStruct

| Type    | Name  | Caption       | Properties                            |
|---------|-------|---------------|---------------------------------------|
| Tstruct | Pbill | Purchase Bill |                                       |
| DC      |       | Header        | TableName = Pbill1,<br>AsGrid = false |

| Type       | Name     | Caption      | Properties  |
|------------|----------|--------------|---|
| InputField | DocNo    | Document No. | DataType = Characters,<br>DataWidth = 10,<br>Mode of Entry = Autogenerate.<br>Click on sequence property and set the values as follows :<br>Prefix = PUR-<br>Description = Sequence for Pbill<br>Starting No = 1<br>Active = True<br>No. of Digits = 6,<br>ReadOnly = true. |
| InputField | DocDate  | Date         | DataType = Date,<br>Mode of Entry = Calculate,<br>Expression = Date().  |
| InputField | Supplier | Supplier     | Mode of Entry = Select<br>Details = From T-struct (Supplier master)<br>Source field = SupplierName,   |
| InputField | SAddress | Address      | Mode of Entry = Fill<br>Details :<br>Master = SupplierName<br>Source = Address,<br>ReadOnly=true  |
| DC         |          | Details      | Table Name = Pbill2<br>AsGrid = true  |
| InputField | Product  | Product      | Mode of Entry = Select<br>Details = From T-struct(Product master)<br>Source field = ProductName,  |
| InputField | Unit     | Unit         | Mode of Entry = Fill<br>Details:  |

| Type       | Name   | Caption | Properties   |
|------------|--------|---------|--|
|            |        |         | Master = ProductName<br>Source = Unit,<br>ReadOnly = true.   |
| InputField | Qty    | Qty     | Mode of Entry= Accept,<br>DataType = Numeric (10,2)<br>Validate Expression =<br>iif(Qty>0, {T}, {Qty should be greater than zero}) |
| InputField | Rate   | Rate    | Mode of Entry = Fill,<br>Details:<br>Master = ProductName<br>Source = PurchaseRate,<br>ReadOnly = true                             |
| InputField | Amount | Amount  | DataType = Numeric (10,2),<br>Mode of Entry = Calculate<br>Expression :<br>Qty * Rate  |

The system creates 2 tables Pbill1 & Pbill2. These have been specified as table names for the 2 DCs in this TStruct. These 2 tables are related through the Pbill1Id. That is the primary table's (Pbill1) primary field (Pbill1id) is created in all the tables related to the TStruct.

The mode of entry for the DocNo field is Autogenerate. So, in the RTE this field will be populated with a sequential number that is generated by the system. The number that is generated will have a 4 character prefix. Click on the sequence property to set the prefix, sequence. (Do not leave the description column empty)

An expression is attached to the DocDate field. The expression will be evaluated & its result will be populated into the field. The user may change this value if needed. In this case, the date function returns the current date. So, the docdate field will be populated with current date suggestively.

Note the properties of Supplier Field.  
Mode of Entry = Select  
Details = From T-struct (Supplier master)  
Source field = SupplierName

This kind of definition will provide a pick list from which the end user can choose. The pick list will contain the suppliernames (from Supplier Master). Instead a reference to the supplier will be stored. That is the primary field of the table in which the supplier name exists will be stored in this field. So, in this case the suppliersid field will be stored in the supplier field. So, a relation ship is

established between the pbill1 table & suppliers table. The product field is also defined as a selectable field.

Note the properties of SAddress field.

Mode of Entry = Fill

Details :

Master = SupplierName

Source = Address,

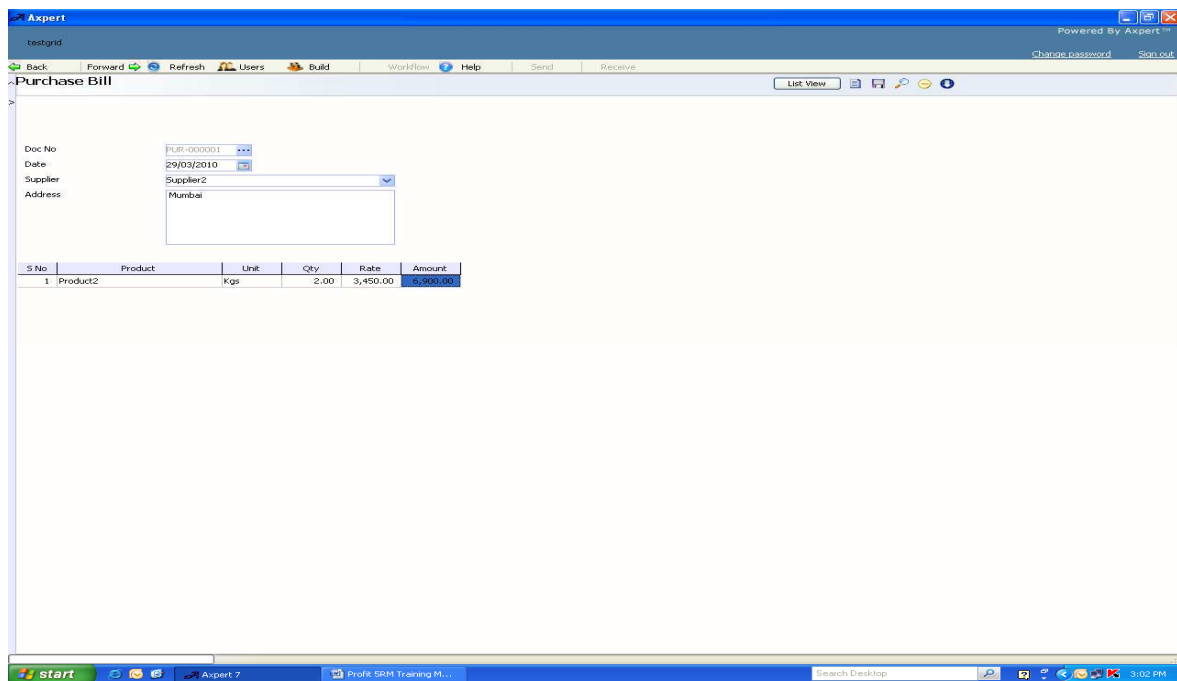
ReadOnly=true

The address (defined as source field) of the selection made in supplier field (defined in Master field) is populated into the Saddress Field.

The Qty field has an expression. Axpert will evaluate this expression and consider the value entered in the field to be valid if the result of this expression is T(true). If the result is F(false), then it will display a message "Invalid value entered in field". If the result is any other string value, the string will be displayed and the value entered will be considered as an invalid value.

The IIF function in the expression takes 3 parameters. If the condition given in the first parameter evaluates to true then the second parameter is returned else the third parameter is returned as the result of the function.

An expression is attached to the amount field. In this case the expression result is populated into the field. However, the user cannot change the value because the mode of entry is "to be calculated".



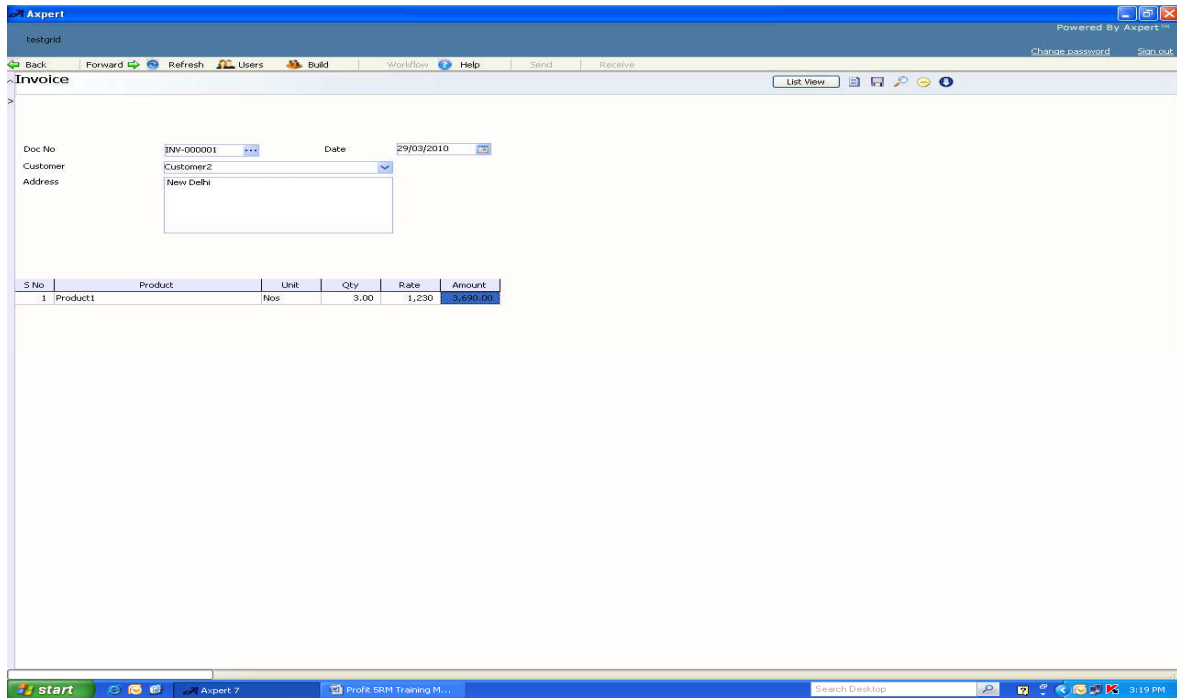
## The Invoice TStruct

| Type       | Name     | Caption      | Properties  |
|------------|----------|--------------|---|
| Tstruct    | Inv      | Invoice      |   |
| DC         |          | Header       | TableName = inv1  |
| InputField | DocNo    | Document No. | DataType = Characters,<br>DataWidth = 10,<br>Mode of Entry = Autogenerate.<br>Details: Prefix = INV-<br>Description = Sequence for invoice<br>Starting No = 1<br>Active = True<br>No. of Digits = 6,<br>ReadOnly= true. |
| InputField | DocDate  | Date         | DataType = Date,<br>Mode of Entry = Calculate,<br>Expression = Date().  |
| InputField | Customer | Customer     | DataType = Charcter,<br>dataWidth = 150,<br>Mode of Entry = Select,<br>Details = From tstruct (CustomerMaster)<br>Source field = CustomerName,  |
| InputField | CAddress | Address      | Mode of Entry = Fill<br>Details:<br>Master = CustomerName<br>Source = Address,<br>ReadOnly = true   |
| DC         |          | Details      | Table Name = Inv2,<br>AsGrid = true   |
| InputField | Product  | Product      | Mode of Entry = Select<br>Details :From t-struct (Product master)<br>Source field = ProductName,  |
| InputField | Unit     | Unit         | Mode of Entry = Fill  |

| Type       | Name   | Caption | Properties  |
|------------|--------|---------|---|
|            |        |         | Master = ProductName<br>Source = Unit,<br>ReadOnly = true.  |
| InputField | Qty    | Qty     | DataType = Numeric (10,2),<br>Validate Expression =<br>iif(Qty>0, {T}, {Qty should be greater than zero})                       |
| InputField | Rate   | Rate    | DataType = Numeric(10,2),<br>Mode of Entry = Fill<br>Master = ProductName<br>Source = SellingRate,<br>ReadOnly = true.          |
| InputField | Amount | Amount  | DataType = Numeric(10,2),<br>Mode of Entry = Calculate<br>Expression =<br>Qty * Rate,<br>AllowEmpty = false,<br>ReadOnly = true |

The user interface & tables will be created as explained in the Purchase bill TStruct.



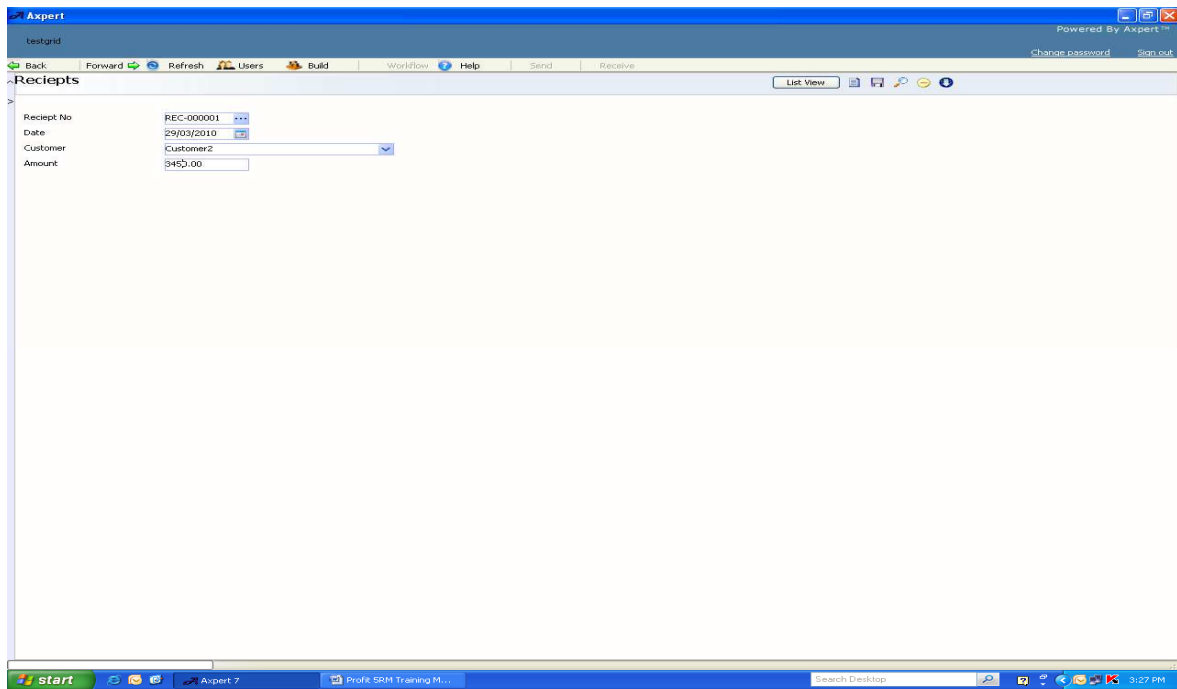


## The Receipt TStruct

This transaction form is intended to accept receipt of money into the company. This accepts the date of receipt, the customer from whom money is received & amount that is received.

| Type       | Name  | Caption    | Properties   |
|------------|-------|------------|--|
| Tstruct    | Rece  | Receipts   |  |
| DC         |       | Details    | TableName = Receipts,<br>AsGrid = false.   |
| InputField | DocNo | Receipt No | Mode of Entry = Autogenerate.<br>DataType = Characters,<br>DataWidth = 10,<br>Mode of Entry = Autogenerate.<br>Click on sequence property and set the values as follows :<br>Prefix = REC-<br>Description = Sequence for Pbill<br>Starting No = 1<br>Active = True |

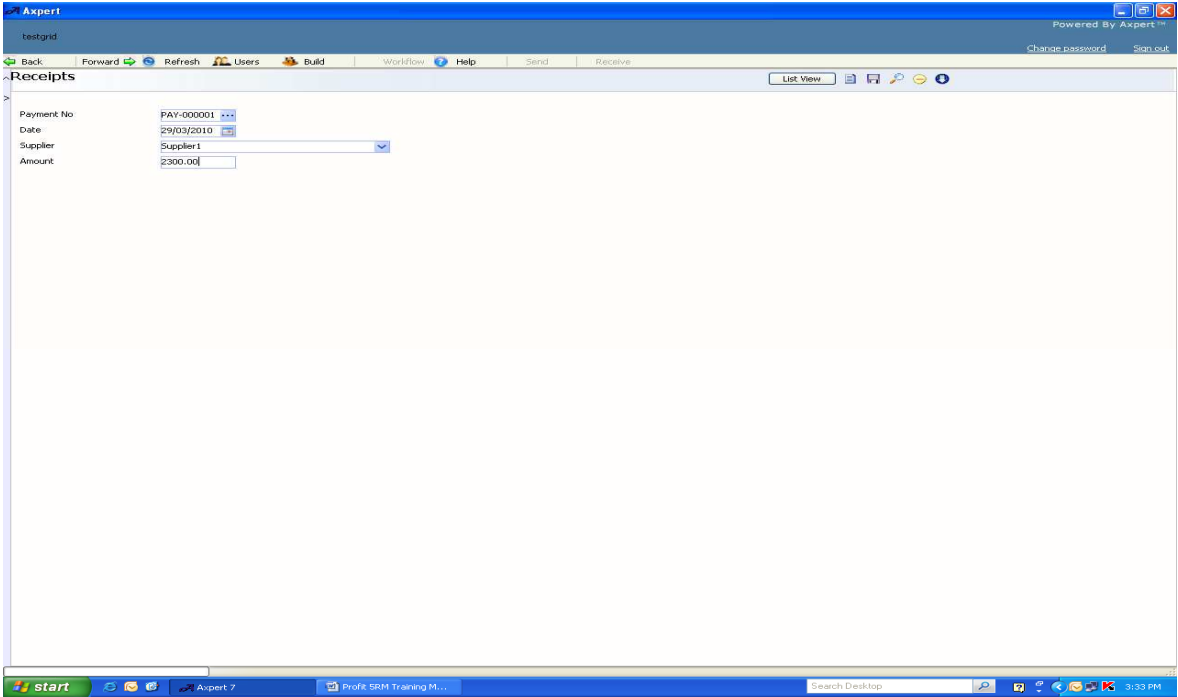
| Type       | Name     | Caption  | Properties  |
|------------|----------|----------|---|
|            |          |          | No. of Digits = 6,<br>ReadOnly = true.  |
| InputField | DocDate  | Date     | DataType = Date,<br>DataWidth = 10,<br>Mode of Entry = Calculate,<br>Expression = Date(),<br>ReadOnly = true. |
| InputField | Customer | Customer | Mode of Entry = Select,<br>Details : From tstruct (Customer master)<br>Source field = CustomerName,           |
| InputField | Amount   | Amount   | DataType = Numeric(10,2),<br>Mode of Entry = Accept   |



### The Payment TStruct

This transaction form is intended to pay money to suppliers. This accepts the date of receipt, the supplier to whom money is paid & amount that is paid.

| Type       | Name     | Caption    | Properties   |
|------------|----------|------------|--|
| Tstruct    | Paym     | Receipts   |  |
| DC         |          | Details    | TableName = payment,<br>AsGrid = false.  |
| InputField | DocNo    | Payment No | Mode of Entry = Autogenerate.<br>DataType = Characters,<br>DataWidth = 10,<br>Mode of Entry = Autogenerate.<br>Click on sequence property and set the values as follows :<br>Prefix = PAY-<br>Description = Sequence for Pbill<br>Starting No = 1<br>Active = True<br>No. of Digits = 6,<br>ReadOnly = true. |
| InputField | DocDate  | Date       | DataType = Date,<br>Expression = Date(),<br>ReadOnly = true  |
| InputField | Supplier | Supplier   | Mode of Entry = Select<br>Details = From tstruct (Supplier master)<br>Source field = SupplierName  |
| InputField | Amount   | Amount     | DataType = Numeric(10,2),<br>Mode of Entry = Accept.   |



### SESSION 3

### MASTER DETAILS & IViews

This session introduces you to Master Detail (MD) Maps & Simple Information Views (IViews). MD Maps are process maps that will update from a detail transaction to a master transaction. IViews are defined to create reports/queries in the application.

Let us extend the application to provide a list of items along with the total qty purchased, total quantity sold & balance as a report. This can be achieved cumulating the qty from purchase bill & invoice into the product master.

Create two more fields in product master to hold the cumulated purchase & sold quantity.

| Type       | Name         | Caption      | Properties                                  |
|------------|--------------|--------------|---|
| InputField | QtyPurchased | QtyPurchased | DataType = Numeric(10,2),<br>Hidden = true. |
| InputField | QtySold      | QtySold      | DataType = Numeric(10,2),<br>Hidden = true. |

Define the following MD Map in Purchase Bill & Invoice

| Type  | Name       | Caption | Properties   |
|-------|------------|---------|--|
| MDMap | UpdateItem |         | Master Transaction = Product master<br>Master Field = QtyPurchased<br>DetailField = Qty<br>MasterSearchField = Product<br>DetailSearchField = Product<br>UpdateType = Add. |

The above definition will add the qty entered in a purchase bill to the QtyPurchased field in Product Master. This will be done whenever a purchase bill is saved, modified, deleted or cancelled.

The product field in purchase bill is related to the ProductId field in the product master. This relationship is mentioned in the Master search field & Detail search field.

To create the report that was outlined in the beginning of this session, you need to create an IView.

| Type         | Name | Properties   |
|--------------|------|--|
| IView        |      | Name = ProdList (max 8 characters allowed, no space)<br>Caption = Products List<br>Heading = List of the Products. |
| Extract Data | SQL  | Select ProductId, ProductName,   |

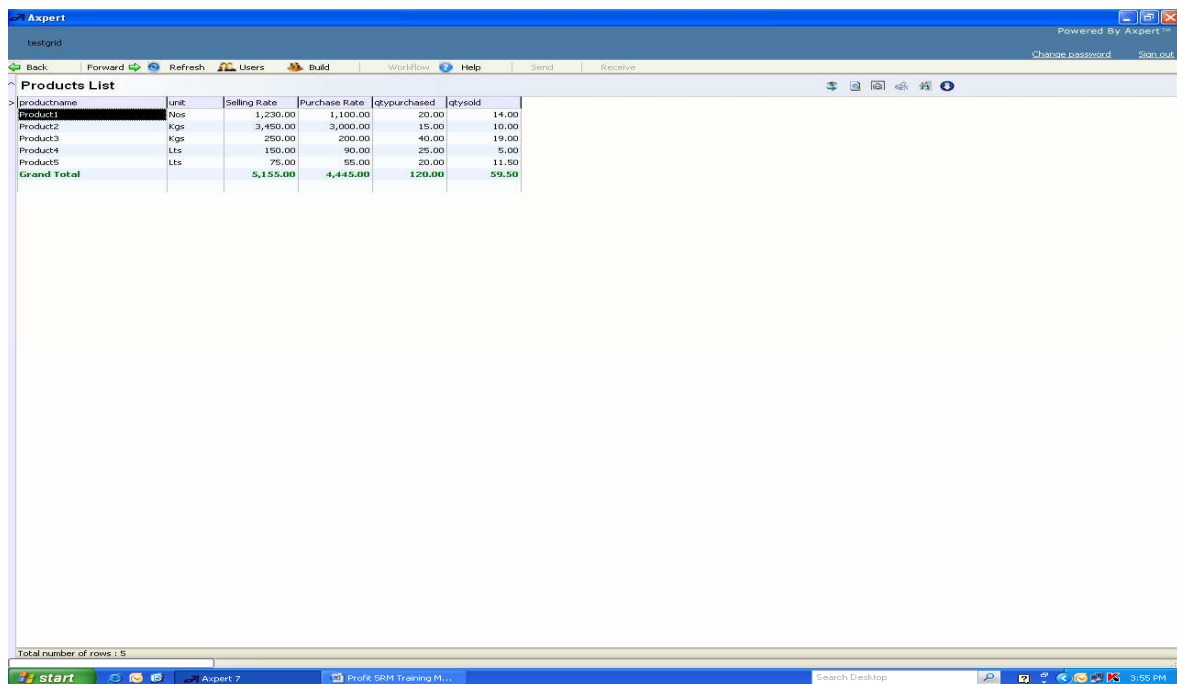
|        |            |  |
|--------|------------|--|
|        |            | Unit,SellingRate,PurchaseRate, QtyPurchased, QtySold from Products;<br>Check := allow sql columns to view. |
| Column | ProductsId | ApplyComma = check,<br>DisplayTotal = uncheck,<br>Hidden = check.  |

Click on the Iview button to create a new IView. Click on the Properties to set the name, caption, heading lines.

Click on the Extract Data button at the bottom panel. The system will present a SQL component. Write the Sql Statement.  
Click on each column & set its properties as shown in the table above.

Save the report & view in the RTE.

Creating a list of invoices sorted customer wise with sub total for every customer. In order to achieve this first add another column in sql statement with CustomerName (with Hidden property in the IView) from Customer Master using where product=ProductsId and customer=CustomerId.  
Click on the button on the bottom panel "Sub Total" which will inturn pop up a window with the column names. Check CustomerName.

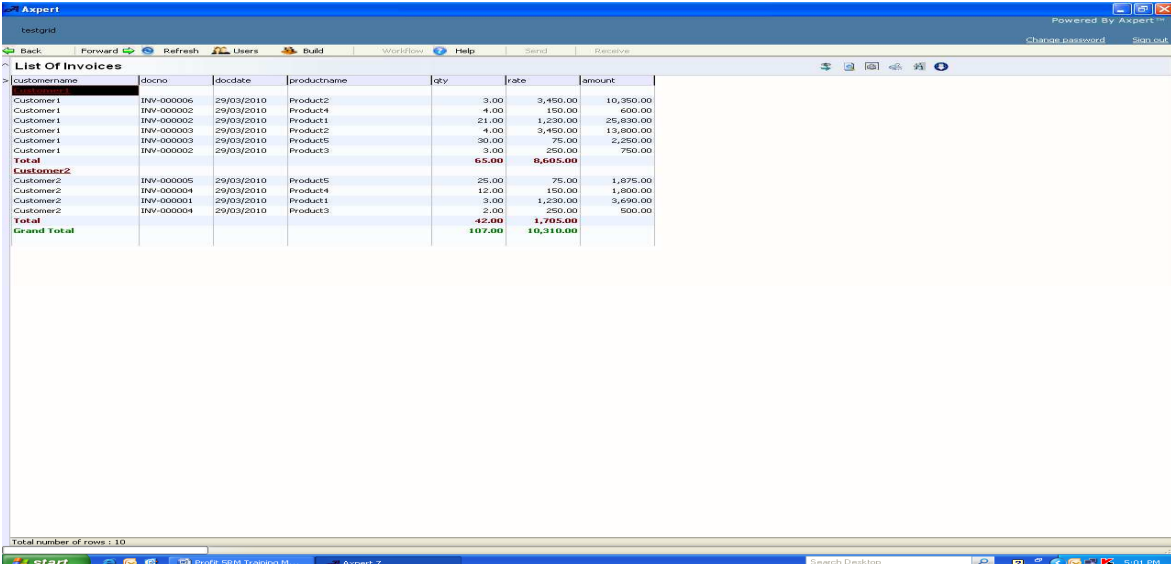


| Type         | Name         | Properties   |
|--------------|--------------|--|
| IView        |              | Caption = List of invoices<br>Heading = List of invoices   |
| Extract Data | SQL          | Select a.inv1id as InvoiceNo, a.docno, a.docdate, c.productname, b.qty, b.rate, amount, d.customername<br>From inv1 a, inv2 b, products c, customers d<br>Where a.inv1id = b.inv1id and a.customer = d.customersid and b.product = c.productsid<br>order by d.customername |
| Column       | Invoice No   | On Double Click on the column. Do the following:<br>ApplyComma = uncheck,<br>DisplayTotal = uncheck,<br>Hidden = check.  |
| Column       | CustomerName | Hidden = true  |

The Block in the above definition will display sub total for every customer. It will print the customer name as a heading.

### Exercise

- ▲ List of purchase bills
- ▲ List of customers
- ▲ List of suppliers



| Customername       | docno      | docdate    | productname | qty           | rate             | amount    |
|--------------------|------------|------------|-------------|---------------|------------------|-----------|
| Customer1          | INV-000006 | 29/03/2010 | Product2    | 3.00          | 3,450.00         | 10,350.00 |
| Customer1          | INV-000002 | 29/03/2010 | Product4    | 4.00          | 150.00           | 600.00    |
| Customer1          | INV-000002 | 29/03/2010 | Product1    | 21.00         | 1,230.00         | 25,830.00 |
| Customer1          | INV-000003 | 29/03/2010 | Product2    | 4.00          | 3,450.00         | 13,800.00 |
| Customer1          | INV-000003 | 29/03/2010 | Product5    | 30.00         | 75.00            | 2,250.00  |
| Customer1          | INV-000002 | 29/03/2010 | Product3    | 3.00          | 250.00           | 750.00    |
| <b>Total</b>       |            |            |             | <b>65.00</b>  | <b>8,605.00</b>  |           |
| Customer2          | INV-000005 | 29/03/2010 | Product5    | 25.00         | 75.00            | 1,875.00  |
| Customer2          | INV-000004 | 29/03/2010 | Product4    | 12.00         | 150.00           | 1,800.00  |
| Customer2          | INV-000001 | 29/03/2010 | Product1    | 3.00          | 1,230.00         | 3,690.00  |
| Customer2          | INV-000004 | 29/03/2010 | Product3    | 2.00          | 250.00           | 500.00    |
| <b>Total</b>       |            |            |             | <b>42.00</b>  | <b>1,705.00</b>  |           |
| <b>Grand Total</b> |            |            |             | <b>107.00</b> | <b>10,310.00</b> |           |

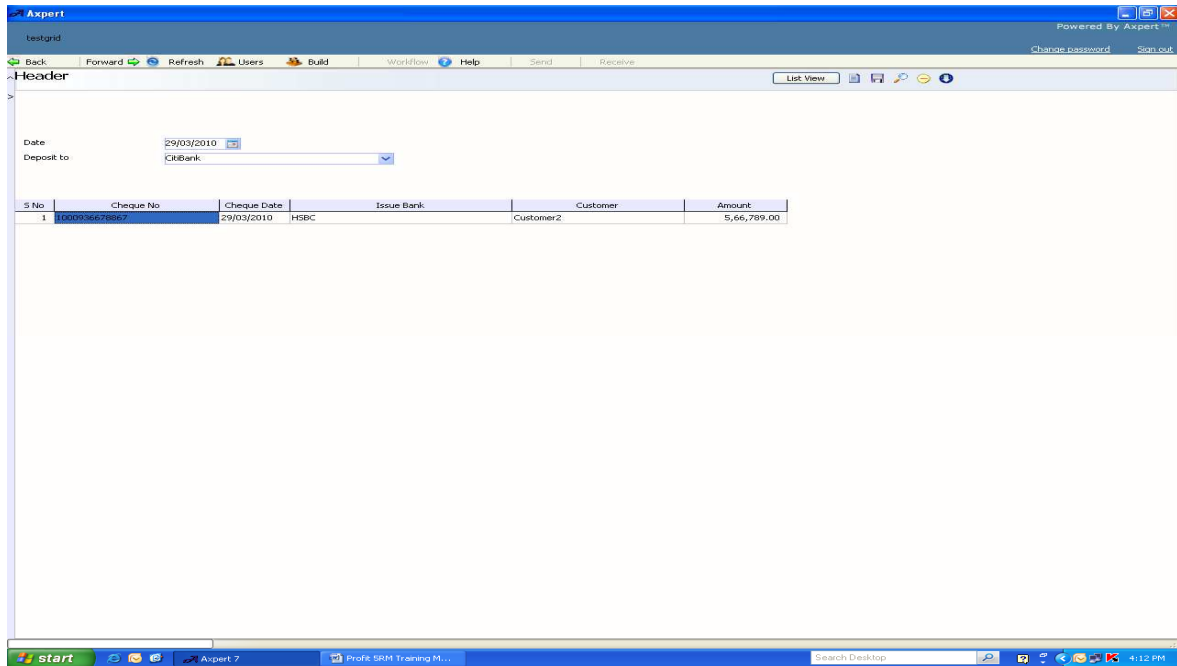
Transaction Generator Maps (GenMaps) are used to post data from one transaction from to another transaction form.

Consider creating a daily cheque deposit form in our application. In this form, the user will enter the date & choose the bank into which deposits are going to be made and gives a list of cheques that have been received on that day. A receipt transaction should be posted for every cheque deposited.

| Type       | Name     | Caption              | Properties   |
|------------|----------|----------------------|--|
| TStruct    | Chqd     | Daily Cheque Deposit |  |
| DC         |          | Header               | Table = chqd1  |
| InputField | DocDate  | Date                 | DataType = Date,<br>DataWidth = 10,<br>Mode of Entry = Calculate,<br>Expression = Date(),<br>ReadOnly = true.                  |
| InputField | BankName | Deposit to           | DataType = Character,<br>DataWidth = 200,<br>Mode of Entry = Select,<br>Details = From List<br>List = "HDFC","Citibank","HSBC" |
| DC         |          | Details              | Table = chqd2<br>AsGrid = true   |
| InputField | Chqno    | Cheque No            | DataType = Numeric,<br>DataWidth = 15,<br>Mode of Entry = Accept.  |
| InputField | Chqdt    | Cheque Date          | DataType = Date,<br>DataWidth = 10,<br>Expression = DocDate,   |



| Type       | Name         | Caption    | Properties   |
|------------|--------------|------------|--|
|            |              |            | ReadOnly = true.   |
| InputField | Bank         | Issue Bank | DataType = Characters,<br>DataWidth = 200,<br>Mode of Entry = Select,<br>Details = From List<br>List = "HDFC","Citibank","HSBC"                  |
| InputField | Customer     | Customer   | DataType = Character,<br>DataWidth = 150,<br>Mode of Entry= Select,<br>Details = From tstruct (Customer Master),<br>Source Field = CustomerName, |
| InputField | Amount       | Amount     | DataType = Numeric<br>DataWidth = (10,2)   |
| GenMap     | Post Receipt |            | Target Transaction = Receipts<br>BasedOnDC = Details<br>Active = True<br>Map :<br>DocDate = docdate<br>Customer = customer<br>Amount = Amount    |



The GenMap defined above will post one receipt transaction for every line entered in the Detail Grid. This is specified by the BaseOnDc property (Details is the name of the grid frame in this TStruct).

If the BasedOnDc property is left empty, then for every source transaction there will be one record posted in the target.

Changes made in the cheque deposit transaction will automatically be carried out in the corresponding receipt transaction.

Now, let us try to enhance this application by handling post dated cheques. A post dated cheque is an entry in the detail grid that has a cheque date greater than the docdate.

In case of post date cheque, the receipt transaction should not be posted. Refer the table below to achieve this.

| Type       | Name         | Caption      | Properties  |
|------------|--------------|--------------|---|
| InputField | PostControl  | Post Control | DataType = character,<br>DataWidth = 1,<br>Mode of entry = Calculate.<br>Expression = iif(chqdt > docdate, {T},<br>{F}) |
| GenMap     | Post Receipt |              | Control Field = PostControl   |

The PostControl field will have a value 'T' if the cheque date is greater than document date.

The posting depends on the value in the field defined as the control field in the GenMap. If this field does not contain 'T', then the row will not be posted into the receipt transaction.

### **Exercise**

Create a transaction structure name 'Stock'. This will contain the following fields

- ▲ DocNo, C(10)
- ▲ DocDate, D(10)
- ▲ ProductName N(15)
- ▲ PlusOrMinus C(1)
- ▲ Qty, N(10,2)
- ▲ Rate, N(10,2)

Post from purchase bill to this transaction structure. Set the PlusOrMinus field to 'p'. Similarly, post from invoice to stock. In this case set the PlusOrMinus to 'm'.

- ❖ *Note Value for PlusOrMinus can be set in the Field Maps by specifying appropriate values in the FieldValue column.*

A FillGrid is a map that provides a way of picking data from database using an SQL statement and populating it into a grid frame.

In the prior session we had created a screen for entering cheques received for the day. Post dated cheques (PDC) also could be entered. The PDCs are not posted as receipts.

Let us extend the Cheque deposit form to provide a feature that will list all the PDCs that need to be deposited on the given date. Allow end user to choose the cheques. Fill the grid frame with the selected cheques. Other cheques for the day can then be entered in the same grid.

Define the fillgrid as follows

| Type     | Name     | Caption | Properties   |
|----------|----------|---------|--|
| FillGrid | Fill PDC |         | SQL = Select chequeno, chequedate, bankname, amount, b.customername from chqd2 a, customers b where postcontrol ='T' and a.customer = b.customersid<br>Map :<br>Chequeno = Chequeno<br>ChequeDt = Chequedt<br>BankName = BankName<br>Amount = Amount<br>Customer = Customer<br>MutliSelect = false |

This fill Grid will fill the grid frame with all PDCs that need to be deposited for that day.

To understand the concept further, let us introduce purchase orders & sales orders into the system now.

Any purchase is made after a purchase order is placed on the supplier. So, let us introduce an input form through which a purchase order can be entered. Purchase bills can then be entered against purchase orders.

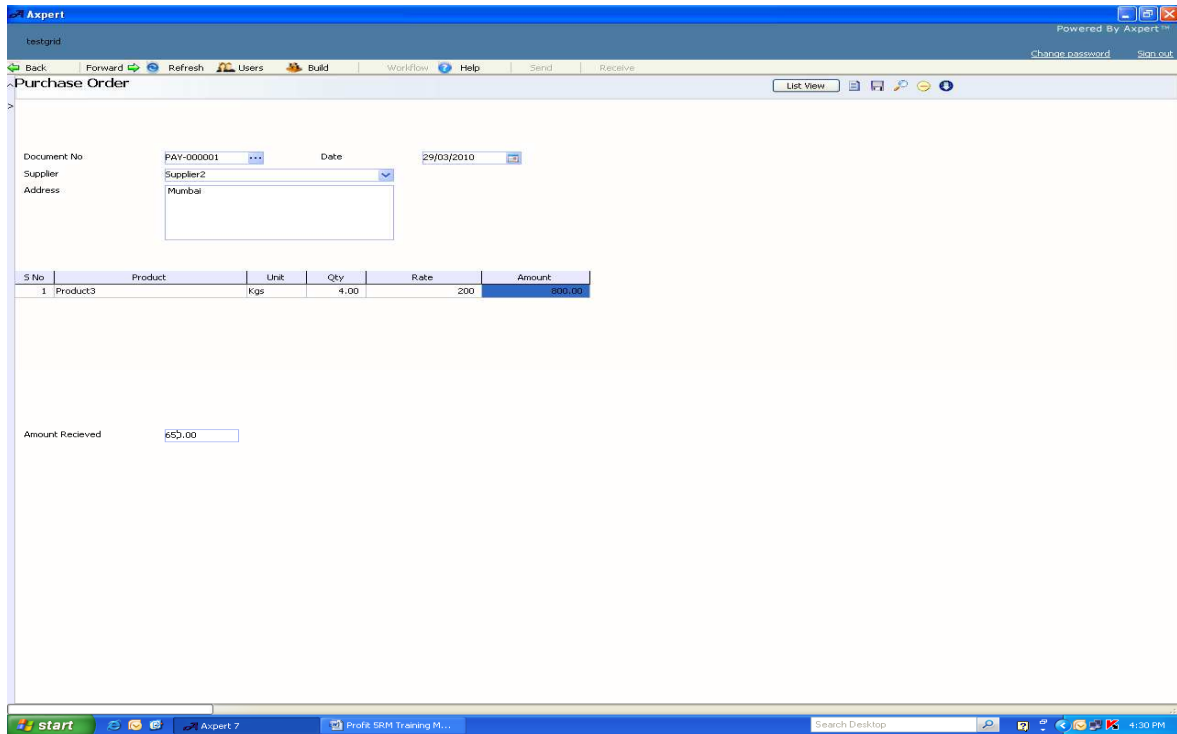
| Type       | Name     | Caption        | Properties  |
|------------|----------|----------------|---|
| Tstruct    | Pord     | Purchase Order |   |
| DC         |          | Header         | TableName = Pord1   |
| InputField | DocNo    | Document No.   | Mode of Entry = Autogenerate. DataType = Characters,<br>DataWidth = 10,<br>Mode of Entry = Autogenerate.<br>Click on sequence property and set the values as follows :<br>Prefix = PAY-<br>Description = Sequence for Pbill<br>Starting No = 1<br>Active = True<br>No. of Digits = 6,<br>ReadOnly = true. |
| InputField | DocDate  | Date           | DataType = Date,<br>DataWidth = 10,<br>Expression = Date().   |
| InputField | Supplier | Supplier       | Mode of Entry = Select<br>Details = From tstruct (Supplier master)<br>Source field = SupplierName   |
| InputField | Address  | Address        | Mode of Entry = From Supplier master<br>Target Key Field = Supplier<br>Source field = Address   |
| DC         |          | Details        | Table Name = Pord2,<br>AsGrid = true  |
| InputField | Product  | Product        | Mode of Entry = Select<br>Details : From Tstruct (Product master)<br>Source field = ProductName   |
| InputField | Unit     | Unit           | Mode of Entry = Fill<br>Details :<br>Master = Product<br>Source = Unit, Read only = True  |

| Type       | Name    | Caption | Properties  |
|------------|---------|---------|---|
| InputField | Qty     | Qty     | DataType = Numeric<br>DataWidth = (10,2)<br>Validate Expression =<br>iif(Qty>0, {T}, {Qty should be greater than zero}) |
| InputField | Rate    | Rate    | Mode of Entry = Fill<br>Master = Product<br>Source = PurchaseRate   |
| InputField | Amount  | Amount  | DataType = Numeric<br>DataWidth = (10,2)<br>Mode of Entry = Calculate<br>Expression = Qty * Rate                        |
| InputField | AmtRecd | AmtRecd | DataType = Numeric<br>DataWidth = (10,2)  |

- ❖ *Note: The Rate field is made suggestive. This means that the value for rate is brought from product master & it can be changed by the end user. If the suggestive is not true, the value brought from the master cannot be changed.*

Make the following change in Purchase bill to bring data from orders.

Now the Purchase bill transaction needs to be changed to pop up all pending orders after the supplier name is selected.



Do the following changes in purchase bill

| Type       | Name    | Caption | Properties   |
|------------|---------|---------|--|
| InputField | Pord2id | Pord2id | N(15)<br>Add this field in the grid frame in Purchase Bill. This will be filled by the fill grid definition. |

|          |             |  |
|----------|-------------|--|
| FillGrid | Fill orders | <p>SQL = Select c.pord2id, b.productname, c.qty -<br/>c.QtyRecd as<br/>qty, c.rate<br/>From pord1 a, products b, pord2 c, suppliers d<br/>Where a.pord1id = c.pord1id and<br/>a.supplier = d.suppliersid and b.productsid =<br/>c.product and d.suppliename = :supplier and c.qty &gt;<br/>c.QtyRecd</p> <p>Map :<br/>Product = Product<br/>Qty = Qty<br/>Rate = Rate<br/>Pord2id = Pord2id<br/>MutliSelect = true<br/>Autoshow = true</p> |
| MDMap    | UpdOrder    | <p>Mater Transaction = Purchase Order<br/>Master Field = QtyRecd<br/>DetailField = Qty<br/>MasterSearchField = Pord2id<br/>DetailsSearchField = Pord2id<br/>UpdateType = Add</p> <p>This MDMap is defined to cumulate the quantity that<br/>has been received in a purchase bill to the<br/>corresponding order.</p>   |



**Exercises**

- ▲ Add a field in the supplier master to hold the total payments made to the supplier. Update this field from the payment transaction using an MD Map.
- ▲ Define the “Payment Report”. This report should list all the suppliers along with the Billed Amount, Paid Amount & Balance.

| Supplier | Amount billed | Amount paid | Balance |
|----------|---------------|-------------|---------|
|          |               |             |         |
|          |               |             |         |

- ▲ Track the payments made bill wise. Allow selection of purchase bill no against which the payment is made in the payment transaction. The selection should provide a list of all pending purchase bills. Some bills may be partly cleared.
- ▲ Define the “Purchase Bills Status” report. This report should list all the purchase bills along with the bill amount, paid amount & pending amount for each supplier.
- ▲ Define the “Purchase bill wise payments” report. This report should list all the bills along with payments. It should look as follows.

| Bill/Pay | Date       | Document No | Amount  |
|----------|------------|-------------|---------|
| Bill     | 01/04/2010 | Bill000001  | 1000.00 |
| Pay      | 02/04/2010 | Pay-000001  | 500.00  |
| Pay      | 03/04/2010 | Pay-000002  | 400.00  |

**SESSION 6**

**IVEIW PARAMETERS**

- ▲ IView Parameters
- ▲ Grouped Views

In session 3, we defined IViews to provide the end user a list of invoices. There may be a need to view the list of invoices between a given start date & end date. In this case, the start date & end date should be accepted as parameters from the end user.

| Type         | Name         | Properties   |
|--------------|--------------|--|
| IView        | invl         | Caption = List of invoices<br>Heading = List of invoices   |
| Extract Data |              | Select a.inv1id, a.docno, a.docdate, c.productname, b.qty, b.rate, amount, d.customername<br>From inv1 a, inv2 b, products c, customers d<br>Where a.inv1id = b.inv1id<br>and a.customer = d.customersid<br>and b.product = c.productsid<br>and a.docdate >= :StartDate and a.docdate <= :EndDate<br>order by d.customername |
| Column       | Inv1id       | ApplyComma = uncheck,<br>DisplayTotal = uncheck,<br>Hidden = check   |
| Sub Total    |              | Check Column = Customername<br>Header caption = {Customername}<br>Footer caption = Total   |
| Column       | CustomerName | Hidden = true  |
| Parameter    | StartDate    | Data type = Date   |
| Parameter    | EndDate      | Data type = Date   |

Notice the SQL statement consists of 2 parameters named startdate & enddate. Parameters are prefixed within an sql statement. AXPERT will automatically add these two as components in the parameters tab. Click on the component to set its properties in the object inspector.

Similarly, a list of invoices can also be provided for a selected supplier. In this case, the supplier name should be accepted as a parameter. Moreover, the supplier name should be selected from a pop up list. Change the IView as follows.

| Type  | Name | Properties   |
|-------|------|--|
| IView |      | Caption = List of invoices<br>Heading = List of invoices   |
| SQL   | SQL1 | Select a.inv1id, a.docno, a.docdate, c.productname, b.qty, b.rate, amount, d.customername<br>From inv1 a, inv2 b, products c, customers d<br>Where a.inv1id = b.inv1id<br>and a.customer = d.customersid |

|           |              |   |
|-----------|--------------|---|
|           |              | and b.product = c.productsid<br>and a.docdate >= :StartDate and a.docdate <= :EndDate and d.customername = :Customername<br>order by d.customername |
| Column    | Inv1id       | ApplyComma = uncheck,<br>DisplayTotal = uncheck,<br>Hidden = check  |
| SubTotal  |              | Blocked Column = Customername<br>Header caption = {Customername}<br>Footer caption = Total  |
| Column    | CustomerName | Hidden = check  |
| Parameter | StartDate    | Data type = Date  |
| Parameter | EndDate      | Data type = Date  |
| Parameter | CName        | Data type = Character<br>SQL = Select customersid, customername from customers order by customername<br>SQLType = ParamSQL                          |

In the above definition the customername parameter is added to the sql statement in SQL1. To provide a pop up list of customers for selection, add another sql of type "ParamSQL" and assign it to the customername parameter. Refer to the properties of SQL2 & CustomerName parameter in the table above.

### **Multi Select in IViews**

Consider a case where there is a requirement to see invoices made to more than one customer in the same report. To implement this, we need the customer to choose more than one customer in the above IView.

This can be achieved by setting the multi select parameter of customername parameter to true.

When a parameter is multi select AXPRT will display a check list of all the customers. The wanted customers can be marked. The selected customers will be stored in a table named axmultiselect. This table has a field name Selection in which the record id of the selected customer (from customers table) is stored. To get a list of invoices of selected customers use the selectedvalues table in the SQL.

### **Iview Defintion Changes**

| Type      | Name         | Properties   |
|-----------|--------------|--|
| Parameter | CustomerName | Mode of Entry = Multi Select   |
| SQL       | SQL1         | Select a.inv1id, a.docno, a.docdate, c.productname, b.qty, b.rate, amount, d.customername<br>From inv1 a, inv2 b, products c, customers d, axpselectionn s |

|  |  |  |
|--|--|--|
|  |  | Where a.inv1id = b.inv1id<br>and a.customer = d.customersid<br>and b.product = c.productsid<br>and a.docdate >= :StartDate and a.docdate <=<br>:EndDate<br>and d.customersid= s.selection<br>order by d.customername |
|--|--|--|

The SQL has been changed to relate to the axpselectionn table. Hence invoices of only selected customers will appear.

### IView definition

| Type         | Name     | Properties  |
|--------------|----------|---|
| IView        | Psum     | Caption = Product Summary<br>Heading = Product Summary  |
| Extract Data |          | Select b.productname as Name1, sum(a.qty) as PurchaseQty<br>from stock1 a, products b<br>where a.productname = b.productsid<br>and a.plusorminus = 'p'<br>and a.docdate <= :AsOnDate<br>Group by b.productname<br>Order by b.productname ;<br>RelationField = Name1 |
| Extract Data |          | Select b.productname as Name2, sum(a.qty) as saleQty<br>from stocktable a, products b<br>where a.product = b.productsid<br>and a.plusorminus = 'm'<br>and a.docdate <= :AsOnDate<br>group by b.productname<br>Order by b.ProductName;<br>RelationField = Name2      |
| Parameter    | AsOnDate | Data Type = Date  |
| Column       | Name1    | Hidden = check  |
| Column       | Name2    | Hidden = check  |
| Column       | Name     | Caption = Product Name<br>Expression = iif(IsEmpty(Name1), Name2, Name1)  |

Note that this definition contains 2 sql statements. The two have the relation field as the productname field. AXPert will display 5 columns. The result set of the 2 SQLs will be joined using a concurrent logic. That is the columns related to the SQL that has the least value in the relation column will be filled. The rest of the column values will be empty.

❖ *Note that the sql result should be ordered on the relation column.*

Assume that sql 1 gives a result as this

| Name1 | PurchaseQty |
|-------|-------------|
| A     | 100         |
| B     | 200         |
| D     | 300         |
| F     | 400         |

Assume that sql 2 gives a result as this

| Name2 | SaleQty |
|-------|---------|
| A     | 50      |
| C     | 100     |
| E     | 150     |
| F     | 200     |

The result of the 2 result sets after relation will be as follows

| Name1 | Name2 | PurchaseQty | SaleQty |
|-------|-------|-------------|---------|
| A     | A     | 100         | 50      |
| B     |       | 200         |         |
|       | C     |             | 100     |
| D     |       | 300         |         |
|       | E     |             | 150     |
| F     | F     | 400         | 200     |

- ❖ Note that the Name1 & Name2 columns are hidden. A new column named name is added. In this expression, it is defined that this column should get filled with the value of name1 or name2 whichever is not empty.

### Grouped Views

Consider making a report from the Invoice tables to display a quarter wise sales of products.

| Product | Q1 Sales | Q2 Sales | Q3 Sales | Q4 Sales |
|---------|----------|----------|----------|----------|
| A       | 100      | 120      | 110      | 150      |
| B       | 130      | 80       | 50       | 400      |

The inv1 & inv2 tables contain the docdate, product, qty fields. An SQL can be used to extract this data from these tables and lay them out in the above format.

### **IView definition**

| Type  | Name | Properties  |
|-------|------|---|
| IView | QSal | Caption = Quarter wise sales<br>Heading = Product sales – Quarter wise  |
| SQL   | SQL1 | Select a.docdate, c.productname, b.qty, month(a.docdate) as monthno<br>From inv1 a, inv2 b, products c<br>Where a.inv1id = b.inv1id |

|        |         |   |
|--------|---------|---|
|        |         | and b.product = c.productsid<br>Order by c.productname  |
| Column | DocDate | Hidden = check  |
| Column | Qty     | Hidden = check  |
| Column | Monthno | Hidden = check  |
| Column | QNo     | ADD Column<br>Expression = iif(MonthNo > 3 & Monthno < 7, 1,<br>lif(MonthNo > 6 & Monthno < 10, 2,<br>lif(MonthNo > 9 & MonthNo < 13, 3, 4)))<br>Hidden = check |
| Column | Q1Sales | ADD Column<br>Expression = iif(QNo=1,Qty,0)   |
| Column | Q2Sales | ADD Column<br>Expression = iif(QNo=2,Qty,0)   |
| Column | Q3Sales | ADD Column<br>Expression = iif(QNo=3,Qty,0)   |
| Column | Q4Sales | ADD Column<br>Expression = iif(QNo=4,Qty,0)   |

In the SQL the month number is arrived at using the Month() function. The QNo column will derive the Quarter number in which the month falls.

Consider an Invoice data as follows

| Invoice Date | Product Name | Qty |
|--------------|--------------|-----|
| 01-apr-2004  | A            | 10  |
| 02-may-2004  | A            | 20  |
| 03-jul-2004  | A            | 30  |
| 06-oct-2004  | A            | 40  |
| 10-dec-2004  | A            | 50  |
| 11-jan-2005  | A            | 60  |
| 31-mar-2005  | A            | 70  |

For the above data, the IView result will be as follows

| Date   | Product | Month | QNo | Q1Sales | Q2Sales | Q3Sales | Q4 |
|--------|---------|-------|-----|---------|---------|---------|----|
| 01-apr | A       | 4     | 1   | 10      |         |         |    |
| 02-may | A       | 5     | 1   | 20      |         |         |    |
| 03-jul | A       | 7     | 2   |         | 30      |         |    |
| 06-oct | A       | 10    | 3   |         |         | 40      |    |
| 10-dec | A       | 12    | 3   |         |         | 50      |    |
| 11-jan | A       | 1     | 4   |         |         |         | 60 |
| 31-mar | A       | 3     | 4   |         |         |         | 70 |

To the above result the GroupField is applied. Note that this is defined as part of the properties in the IView.

In a grouped view, all the rows that contain the same value in the group column will be consolidated into one row. When consolidating, all the numeric columns will be summed & displayed. The value in the first row will be displayed for all non numeric columns.

Hence the result of the above IView after grouping will be

| Date   | Product | Month | QNo | Q1Sales | Q2Sales | Q3Sales | Q4  |
|--------|---------|-------|-----|---------|---------|---------|-----|
| 01-apr | A       | 42    | 18  | 30      | 30      | 90      | 130 |

The needed report can be arrived at by hiding all the other columns except the Product & the Q1..Q4 columns.

### Drill Down

Most software applications provide features to drill down from a summary to a detail level. For example, it will be a good feature to provide customer wise summary of sales. On double click of any customer, the list of invoices for the customer should appear.

### **IView definition for Invoice Details**

| Type         | Name         | Properties  |
|--------------|--------------|---|
| IView        | invd         | Caption = Invoice Details<br>Heading = Invoice Details  |
| Extract Data |              | Select a.inv1id, a.docno, a.docdate, c.productname, b.qty, b.rate, amount, d.customername<br>From inv1 a, inv2 b, products c, customers d<br>Where a.inv1id = b.inv1id<br>and a.customer = d.customersid<br>and b.product = c.productsid<br>and d.customername = :customername<br>order by d.customername |
| Column       | Inv1id       | ApplyComma = Uncheck,<br>DisplayTotal= Uncheck,<br>Hidden= check  |
| SubTotal     |              | Blocked Column = Customername<br>Header caption = {Customername}<br>Footer caption = Total  |
| Column       | CustomerName | Hidden = check  |

### **IView definition for Customer summary**

| Type         | Name | Properties   |
|--------------|------|--|
| IView        | Csum | Caption = Customer summary<br>Heading = Customer summary |
| Extract Data |      | Select customername, qtysold                             |

|        |             |   |
|--------|-------------|---|
|        |             | from customers<br>order by customername |
| Action | Inv Details |   |

When the next button is clicked, the Invoice Details Iview will be displayed. This IView accepts customername as a parameter. Since there is a column named customername in Customer summary iview, the value in this column in the current row is sent as a parameter to the details iview. So, only the invoice details of the selected customer are displayed in that Iview.



**SESSION 7**

**Sub forms (PopUP Grid)**

Sub forms are used to accept multiple rows of data about each row in a grid DC. This could be used to establish a many to many relationship between the grid DC and the sub form.

Consider a purchase bill that accepts a list of items purchased in a grid DC. Each item in the grid has an associated list of serial numbers. This data needs to be accepted. Hence, the system should pop up a window for every line in the item details grid & accept a list of the serial numbers.

Change the Purchase Bill as given below.

| Type       | Name        | Caption       | Properties   |
|------------|-------------|---------------|--|
| TStruct    | PBill       | Purchase Bill | This TStruct exists already. Make necessary changes.   |
| DC         | SerialNoDC  | Serial Nos    | Table Name = PBill3<br>AsGrid = True<br>DC Type = SubGrid<br>Invoke Field = Qty<br>Invoke DC = Details<br>Validate Field = SubQty<br>Validate Exprn = Qty<br>IdField = Product |
| InputField | SerialNo    | Serial No     |  |
| InputField | Description | Description   | DataType = Character<br>DataWidth = 10   |
| InputField | SQty        | SQty          | DataType = Numeric,<br>Datawidth = 1,<br>Expression = 1,<br>Hidden = Check   |

A new DC is created in the PBill TStruct. The DCType property specifies that this is a sub form. The form will pop up whenever the value in the Invoke Field changes. In this case, when the user enters some value in the qty column, the form pops up. This form shows a grid with the serialNo & Description as columns. These have been defined as input fields in the DC.

The form is based on the Details DC. This is defined by setting the Invoke DC property. AXPERT will establish a relationship between the sub form table (pbill3) and the details grid table (pbill2). The parentrecordid field in pbill3 will contain the pbill2id of the pbill table. The validate field is a field in the sub form DC. The total value in this field will be validated against the result of the validate expression. In this case the system will ensure that the total value in the SQty value does not exceed the value given in the qty column in the details grid.

The IdField is a parent field to the sub form data. That is if the value in the id field is changed after data is entered in the sub grid, the corresponding data related to that row will automatically be removed by the system.

This feature can be used wherever there is a need to split the data in a grid DC into further rows.

Refer Axpert – Actions. Doc in Training folder.. (Separate files)

For more details refer the below documents: Axpert – Commands, Axpert – Tasks, Functions in Axpert.

### Controlling saving

It is important to control saving of a transaction entered by the user. AXPERT will consider a transaction to be invalid if any of the validation expressions defined for the input fields returns “F”. Invalid transactions will not be stored into the database. Further to this, any field in the TStruct can be defined as the save control field. If this does not contain T then the transaction is invalid. The save control field is set as a property for the TStruct.

Consider implementing rules as follows

- ▲ Modifying a purchase order is not allowed after a purchase bill has been entered against the purchase order.
- ▲ Modifying the purchase bill is not allowed after a payment is made against the bill.

To implement the control in a purchase order, do the change in the Purchase order TStruct as given below. Add the given fields in the footer DC.

| Type       | Name        | Caption     | Properties  |
|------------|-------------|-------------|---|
| InputField | BillExists  | BillExists  | Mode of entry = Select<br>Details = From SQL<br>SQL = select DocNo<br>from prod1 a, pord2 b, pbill2 c<br>where a.pord1id = b.pord1id<br>and b.pord2id = c.pord2id<br>and a.pord1id = :recorded;<br>Hidden = Check |
| InputField | SaveControl | SaveControl | Mode of Entry = Calculate<br>Expression = iif(isempty(BillExists), {T}, {F})  |

Select the SaveControl in the Save Control field property. In the SQL defined for this field, the parameter recordid is a system var. This will give the record id of the primary table of the transaction that is loaded for modification. In entry mode this will be 0. The SQL gets a the DocNo

of the purchase bill that has been entered for this record. The SaveControl field will result in T if only if BillExists is empty. So the transaction will be saved only if no bill exists for this purchase order.

### **Controlling deletion**

Similar to controlling saving of transactions, AXPERT also allows controlling of deletion of transactions. A delete control field can be set for every TStruct.

The rules that are applicable for modification of purchase orders and purchase bills are also applicable for deletion. Hence the SaveControl can be set as a Delete Control field in the Purchase Order TStruct.

### **Print formats**

Refer to the Axpert developer manual to understand print formats.

*Exercises:*

- ▲ Create a print format to print invoices.
- ▲ Create a print format to print purchase orders.
- ▲ Create the print format through MS Word.