Accounting System Using Database Management System

15

LEARNING OBJECTIVES

After studying this chapter, you will be able to :

- *identify the resources of MS ACCESS as DBMS;*
- *create data tables described in a database design and set relationship among these tables;*
- *explain the ACCESS basics and procedures to create forms using ACCESS;*
- *describe and create voucher forms in consonance with different database designs;*
- *identify information requirement of reports for querying databases;*
- *formulate and implement queries for retrieving data and information for presentation in accounting reports ; and*
- *implement the process in ACCESS for generating accounting reports by using accounting information queries.*

In chapter 14, you have learnt about the fundamentals of creating a database design in the context of accounting system. This chapter deals with the basics of MS Access for implementing the databases and specifically deals with implementation of accounting databases, the design of which has been shown, described and discussed in chapter 14 as Model-I and Model-II. The accounting database design has been discussed below in terms of its implementation modalities in the context of MS Access.

15.1 MS Access and its Components

It is one of the popularly used Database Management System (DBMS) to create, store and manage database. It is also popularly called ACCESS.

Every component that is created using Access is an object and several such similar objects constitute a class. Access is functionally available with the following seven-object classes. Each of these object classes is capable of creating their respective object replicas.

- *Tables* : This object class allows a database designer to create the data tables with their respective fieldnames, data types and properties.
- *Queries* : This object class is meant to create the SQL compatible query statement with or without the help of Graphic User Interface (GUI) to define tables, store data and retrieve both data and information.

- *Forms* : This object class allows the designer to create an appropriate user interface to formally interact with the back end database, defined by the tables and queries.
- *Reports*: This object class is used to create various reports, the source of information content of which is based on tables, queries or both. Such reports are designed in Access according to the requirement of end-user.
- *Pages* : This object class is meant to create Data Access Pages, which can be posted on a Web site of an organisation using Internet or sent via e-mail to someone of the organisation's network.
- *Macros* : In macro programming, the objects using individual instructions called macro-oriented actions are manipulated. A Macro is a list of macro-oriented actions that run as a unit. Access provides for such Macro programming.

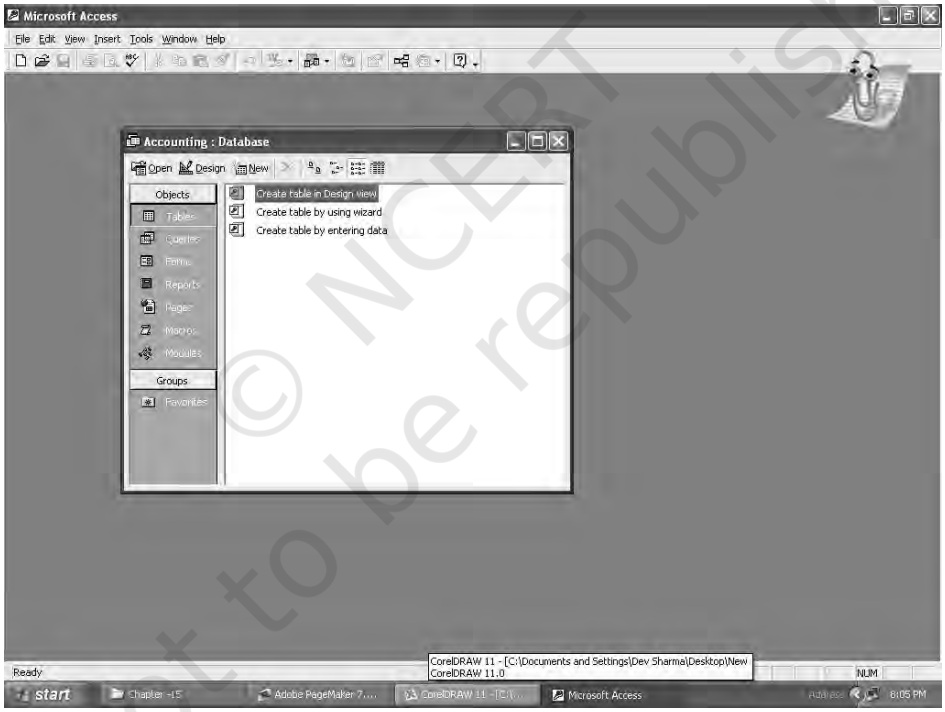


Fig. 15.1 : *An example of database window to work in Access*

- *Modules* : These are the foundations of any application and allow the designer to create a set of programming instructions, called functions or sub-routines that can be used throughout the application.

The functions return a value while subroutines do not return any value. Access provides for creating such modules.

Each of these object classes is contained in the named database file of Access with ***MDB*** extension. Whenever this file is opened, a database window, as shown on next page, opens with all the above object classes available on the left hand side. As and when the specific objects are created or designed, they get listed on right hand side of this window against each of these object classes.

Box 1
Capabilities of MS Access

Access has certain capabilities, which bring it closer to an ideal Database Management System. These capabilities are :

- Storing the data in an organised manner.
- Enforcing data integrity constraints.
- Representing complex relationship among data.
- Providing for persistent storage of database objects.
- Restricting unauthorised access to database.
- Allowing fast retrieval of data with or without processing by using SQL.
- Flexibility to create multiple user interfaces.
- Providing for data sharing and multi-user transaction processing.
- Supporting multiple views of data and information.

15.1.1 Access Basics for Creating a Database

When a new database is created from the scratch, there is complete control over the database objects, their properties and the relationships. In order to create a new database without the help of database ***wizard*** (that is an automated process in Access), the following steps are required :

(i) Open Access Window to choose blank Access database and click OK button.

(ii) Access responds by displaying File New Database dialog box, which prompts the designer to enter a file name and a location for the database. This must be followed by clicking ***Create*** button.

(iii) If the task pane is not open, choose ***File*** from menu bar and click at ***new*** to open the task pane to create a new database.

15.1.2 Creating of Tables in Access

The creation of tables in Access requires the following steps and understanding of the components of ***table*** object.

Click at ***Tables*** object of Access, followed by double click at create ***table by design view***. This results in providing a table window, the upper part of which has three columns: Field Name, Data Type and Description. It is meant to define the schema of a table being created. Each of its rows corresponds to a column of the table being created. Two primary properties of the column of a table are its field name and data type.

(a) *Field name* : refers to column name of the table being created. The name of the column should be a string of contiguous characters. The Field name is meant to define the name of column to be created, followed by data type of such column. The designer can optionally provide description of the column also. Once the data type is defined, the designer can further specify the properties of each column in the lower part of the Table window.

(b) *Data Types* : Access supports different data types, the details of which are as given below :

- *Text* : It is used for a string of characters: words or numbers that are not to be used in any arithmetic calculations. The maximum length for a text field is 255 characters. It is the default data type because of being used most frequently.
- *Memo* : It is used for storing comments and is capable of accommodating 65,536 characters. But a field with this data type is not amenable to sorting or filtering of data records.
- *Number* : It is meant to store numbers, which could be integers (-32768 to 32767), long integers (–2,147,483,648 to 2,147,483,647), bytes (0 – 255), single (to store values with decimal point up to a certain limit), double (to store values in decimal point with greater magnitude and more precision) or decimal types.
- *Date/Time* : It is used to store dates, times or a combination of both.
- *Currency* : It is used for storing numbers in terms of Dollars, Rupees or other Currencies.
- *AutoNumber* : It is a numeric data automatically entered by Access. It is of particular importance in a situation where none of the fields individually or a set of fields as a combination in a table is unique.
- *Yes/No* : It is to declare a logical field which may have only one of the two opposite values alternatively given as: Yes or No, On or Off, True or False.
- *OLE Object* : OLE stands for Object Linking and Embedding. It refers to an object that could be a photograph, bar code image or another document created in another software application.
- *Hyperlink* : This data type is meant to store a Universal Resource Locator (URL) and e-mail addresses.

(c) *Properties* : Once the data type of a column is specified, Access allows the designer to define the properties of each column. These properties are of two types General and Look up.

(i) *General* : In the context of text data type the general properties are :
- *Field Size* : This property, in case of text fields, refers to the maximum number of characters allowed in the column. The same property, in case of numbers, refers to the type of numbers being stored as per requirements.
- *Format* : It is meant to indicate as to how the field's contents are displayed. There are standard types of formats to choose from.
- *Decimal places property* : It applies to single, double or decimal types of numbers.
- *Input mask* : Formats for data entry that include placeholders and punctuations are called input masks. It works only for text and date type of fields. It is of particular importance when the accounting codes being used in the system are formatted with hyphens.
- *Caption* : It is a label used for the field in datasheet view and on the Forms and reports. If the caption property is set to blank, the field name becomes the default caption and is used to label the field.
- *Default Value* : It is used for specifying a value for new entries of data records. While entering the data item, the operator can always over write the default value. The default value should be the most frequently entered value in the field.
- *Validation Rule and Text* : Validation means checking of data to eliminate incorrect entries. Validation criteria can be specified for this property. If the data so entered does not satisfy the validation criteria, the validation text gets displayed.
- *Required and Indexed* : The Required property must be provided a logical value ***Yes*** or ***No***. When a field's required property is set to ***Yes***, a user must enter data in the field before saving the record. A value of ***No*** implies that the data entry in the field is optional. In other words, a null value is also acceptable to the database. Indexing a field results in speeding up sorting, searching and filtering of records on that field. Primary key field is always indexed. For a single field primary key, Access sets the ***Required*** property to Yes and the Indexed property to Yes (No duplicates) because a primary key by definition must have unique values without null entries.
- *Allow-Zero Length* : This property is available only for text fields. Setting it to Yes/No determines whether a text string with zero length is a valid entry or not.

(ii) *Look up* : The look up feature is used by a field to find its values in another table, query or from a fixed list of values. A list of valid values can be displayed using a list box or combo box. Text box is the default display control of look up. Look up is created in case of a field, which is foreign

key (many side) into primary key (one side) between the tables that have one-to-many relationship. Its other display controls are list control and combo control. When list box or combo box is used as display control in look up, it is important to specify the row source type (that is table, query or list of values or field list). The list of values must be separated by comma. Some additional properties in case of list box or combo box are meant to specify the bound column whose values are copied to this field as references. Number of columns to appear in the list box or combo box is determined by column count property.

- The above steps for defining a column need be repeated for every column to be created for a particular table.
- After defining all the columns of the table, the primary key column of the table can be specified as any of the columns that are expected to have unique data values. This can be achieved by right clicking at the field to be specified as primary key followed by primary key item of right clicked window. If more than one field constitutes a primary key, select first field (of such composite primary key) by pressing and holding Ctrl key and clicking other fields (of the composite primary key) one by one in the same order in which they together constitute the primary key. This must be followed by right click at selected fields to mark the selected fields as primary key.
- Save the table design by clicking at ***File*** item of menu bar followed by click at ***Save*** option. Access responds by providing a generic default name of table. The table name provided by Access may be accepted by clicking at OK or changed by re-typing another name at the input dialog box. This must be followed by clicking OK. The table stands created and appears as listed to the right of table object.
- Every other table, which constitutes part of the database design, may also be created in the same manner as described above.

The foregoing discussion in this chapter is divided into four sections: Creating tables and relationships for accounting databases; Vouchers and forms; information using queries and generating accounting reports.

15.2 Creating Tables and Relationships for Accounting Database

The database designed in of this chapter is to be discussed in the context of database components as detailed above. This is because the implementation of each database design is conditioned by its particular table structure and interrelationships. Such implementation modalities have been discussed in detail for various types of transaction vouchers already described in the preceding chapter.

15.2.1 Database Design for Simple Transaction Vouchers

According to the design shown in figure 14.24 (Model-1) of preceding chapter**,** there are ***five*** data tables: Employees, Accounts, Vouchers, Support and AccountType. For the purpose of implementation, each table is described below in terms of their storage structure, i.e. column names, data types and properties:

(a) AccountType : This table has two columns: CatId and Category.

- *CatId* : This column of the AccountType table is meant to specify the identification value of the category of accounts. Since there are limited number of accounts type and are being expressed as numeric only, the data type of this field can be safely taken as 'Number/byte' because the storage space taken by the data type 'Number/byte' is minimum. This field has been designated as primary key because it has unique values across a set of category records.
- *Category* : This field is meant to store the string of characters to express the category of account such as Expenses, Revenues, Assets and Liabilities. Its data type should be ***Text*** with suggested field size set to 15 characters.

(b) *Accounts* : This table has three columns: Code, Name and Type.

- *Code* : A unique account number or code identifies an account**.** This column is meant to store this code. Its data type is chosen as ***Text*** because it is not to be subjected to any calculations. Its field size is required to have a length of six characters because every account is designed to have six digits at leaf level. Because of uniqueness in values, this field is a good primary key field. The ***Allow Zero Length*** property must be set to ***No. Indexed*** property of this field must be set to ***Yes*** (No duplicates) to imply that the database creates automatically an internal index on this field for fast retrieval of data records and ***No duplicates*** indicates that this index is based on unique values of code.
- *Name* : In a system of accounting, every account has a name. This column is meant to store the name of an account corresponding to the account code by which it is identified. Its data type is declared as ***Text*** because it is a string of characters not required for any calculations. Its field size need be set to 30 characters, which is considered to be long enough to accommodate the name of account.
- *Type* : Every account must belong to one of the accounts type as stored in AccountType table. This field is a foreign key to reference CatId field of AccountType table. Its data type and other properties must be the same as that of CatId field in AccountType table, except that its Index property can be set to YES (Duplicates OK). This is because Type value within accounts table cannot be unique as a number of accounts might belong to a particular AccountType and store a common CatId

as data value in Type field. The relationship between the CatId column of AccountType table and type column of Accounts table must also be defined so as to maintain referential integrity.

(c) *Employees* : This table stores the data pertaining to employees of the organisation and is designed to have following columns :

- *EmpId* : Each employee is identified by a unique data value called EmpId, which in turn gets reflected in employee table as a column to store for each employee record a unique identification value. The data type of this column is text with field size equal to 4. Being a column to store unique values and also because of its capability to identify an employee record, it is designated as primary key field. Its ***Required*** property is set to ***Yes*** and ***Zero length property*** is set to ***No*** with ***Indexed*** property as ***Yes*** (No Duplicates).
- *Fname* : This column refers to the first name of employee and its data type is declared as Text because it is meant to store string of alphabets. Its Field size is set to 10 on the assumption that first name of every employee can be completely accommodated within this field size. The ***Required*** property is set to Yes with ***Zero Length Property*** being No to imply that every employee has a first name and no employee record can be stored unless the first name is also stored.
- *Mname* : Mname column is meant to store the middle name of an employee. It data type is declared as ***text*** with field width equal to 10. The ***Required Property*** can be set to ***No*** and Zero ***Length property*** to ***Yes*** to imply that many employees may not have middle name. Therefore, the storing of value in this field becomes optional.
- *Lname* : Lname column has been included in the table structure to store the Last name of an employee. The data type of this column is Text with field size set to 10. The Required Property can be set to ***No*** and ***Allow Zero Length*** property to Yes for the reason which applies to Mname.
- *PhoneNo* : This column is meant to store the Phone number of the employee and its data type is set to ***Text*** with field size equal to 12. The Required property is set to ***No*** with ***Allow Zero Length*** property set to Yes to imply that null values are permitted for this field because many employees may not have phone numbers.
- *SuperId* : This column in the Employee table structure refers to EmpId of the supervisor or immediate superior of the employee. Its data type is set to Text with field width 4, the same as is for EmpId. Its ***Required*** property is set to ***No*** with ***Allow Zero Length*** property being ***Yes*** to imply that null values are also permitted. This is because the overall boss of the organisation, although an employee, does not have any

boss and therefore a null value in this field is to be allowed to accommodate the situation.

(d) *Vouchers* : This table has been designed to store the transaction data as contained in a voucher. It has nine columns, the details of each are given below :

- *Vno* : This column is meant to store voucher number, which indicates the distinct identity of a transaction. Its data type could be number if numeric digits are assigned to each of the vouchers. However, its data type is normally taken as text because it is amenable to any type of numbering, coding or ordering scheme: numeric, alpha-numeric or formatted reference. Its width may be set to 6 so that first 2 places to the left refer to numeric month of the date and next 4 places to numeric digits giving identity to each of the transactions that have occurred during the month under reference. This column is designed to have distinct values and therefore can be designated as primary key of the table. Accordingly, its value cannot be null and therefore its ***Allow Zero Length*** property must be set to ***No*** with ***Required*** property being ***Yes***. However, its data type needs be taken as number (with Integer), when numeric function(s) such as Dmax() is applied to find maximum values for auto-generating the vouchers.
- *Debit* : This column is meant to store the code corresponding to an account, which has been debited in recording a transaction. Since it references the ***code*** column, which is the primary key of ***Accounts*** table as described above, it is a foreign key column in ***Vouchers*** table. The data type and properties of this column should be the same as that of code column of Accounts table, except that its ***Indexed*** property need be set to ***Yes*** Duplicates OK). The relationship between the code column of accounts table and debit column of Vouchers table must also be defined so as to maintain referential integrity.
- *Amount* : This column is meant to store the amount of transaction and is common to the accounts being debited and credited. Its data type can be Number with field size set to double; format set to standard; decimal places set to 2 and default value set to 0.00. Alternatively, its data type can be chosen as currency type, in that case its format can be either accepted as currency or set to standard with decimal places set to 2.
- *Vdate* : This column of the table stores the date of transaction. Its data type is set to ***Date/Time*** with format set to Medium Date (dd-MMM-yy); Default value set to = Now() to imply current date in Real Time Clock (RTC) of computer system and caption property set to Date.

- *Credit* : This column is meant to store the code corresponding to the account being credited in recording a transaction. Like Debit column, this column too shares the same properties as code column of Accounts table and must also be dealt with in the same manner as Debit column described above.
- *Narration* : This column is meant to store the narration. Its data type can be set to text type with field size set to 100 characters; Required to ***No; Allow Zero Length*** to ***Yes*** and Indexed to ***No***. If the narrations are very large beyond 255 characters, its data type can be set to ***Memo*** so as to accommodate the narrations up to 65,536 characters, almost equal to 64 pages.
- *PrepBy* : This column is meant to store the identity of an employee who has prepared the voucher. EmpId as defined and described in schema of Employees table identifies the employee. The data type of this field and other properties must be identical to that of EmpId, except that its Indexed property must be set to ***No***. This column as per design is expected to refer to EmpId column of ***Employees*** table and therefore must be defined as foreign key. Its relationship with EmpId column of Employees table must also be specified to ensure referential integrity.
- *AuthBy* : This column is meant to store the identity of the employee who has authorised the vouchers. This column is similar to ***PrepBy*** column. Therefore, its data type, properties and relationship with EmpId are the same as those for ***PrepBy*** column.
- *Support* : This table is created to store the details of support documents annexed to a voucher. It is designed to have the following four columns:
  - *vNo* : This column is meant to store the voucher number to which this document is annexed. Its data type should be the same as that of Vno in Vouchers table because this column refers to Vno column of Vouchers table to maintain referential integrity. Its value cannot be null and therefore its ***Allow Zero Length*** property must be set to ***No*** with Required Property being ***Yes***. Since there may be more than one support documents annexed to a voucher, the values stored in this column cannot be unique and therefore this column alone cannot be a primary key field.
  - *sNo* : This column has been included in the table structure to store serial numbers 1,2,3... to correspond to the serial number of documents being annexed. Duplicate values will occur in this field also because the serial number of documents across the vouchers shall be the same. However, both the columns: Svno and Sno together provide a unique value because the documents, for every voucher are serially numbered and therefore unique. Both the

columns together need be declared as Primary key of this table.

- *dName* : This column refers to Document name. Its data type is ***Text*** with field size equal to 30 to mean that within this character limit the document name can be suitable accommodated.
- *sDate* : This column refers to any date reference given in the support document. Its data type is ***Date/Time***. Its format can be declared as Medium Date with Required and ***Indexed*** property set to ***No***.

15.2.2 Modified Design for Implementing Compound Vouchers

There are two tables: VouchersMain and VouchersDetails

(a) *VouchersMain* : This table has been created to store one record for every transaction. The rows of this table refer to those data items of the vouchers, which lie outside the voucher grid. It consists of Vno, AccCode, vdate, PrepBy, AuthBy and Type.

- *AccCode* : This column is meant to store the complementing account code, which in the context of debit voucher is credit account and in the context of credit voucher is a debit account. In debit voucher, the debit accounts are displayed in Debit Accounts Grid and therefore the complementing account is the account to be credited. Similarly, in Credit Voucher, the Credit Accounts Grid displays only the accounts, which are being credited in recording a transaction. Therefore, the complementing debit account need be stored in this column. This column is also the foreign key column because it references the primary key column of Accounts table. Its data type and properties must be the same as that of Code column of Accounts table, except that its Indexed property must be set to ***Yes*** (Duplicates OK) and the domain of its data values is confined to the code values stored in Accounts table.
- *Type* : This column has been created to store a value **0** (for debit voucher) or **1** (for credit voucher). Its data type therefore is set to ***Number*** with field size set to byte. This column is very important and therefore its values must be carefully stored and interpreted in preparing accounting reports. Improper handling of this column may cause the Errors of Principle in accounting. The data types and properties of Vno,Vdate, AuthBy and PrepBy continues to be same as have been defined and discussed in Vouchers table of Simple Vouchers Design. However, Vno column has acquired an added importance because of being referenced by Vno column of VouchersDetail table.

(b) *VouchersDetail* : This table is meant to store those data items of the voucher, which appear in the grid of debit or credit vouchers. However, the Total

amount of voucher is not stored because it is derived data. It consists of Vno, Sno, Code, Amount and Narration as its columns.

- *Vno* : This column is meant to store voucher number of Debit/ Credit record of VouchersMain table to which the Credit/Debit entries of vouchersDetails table are related. Its data type should be the same as that of Vno in VouchersMain table because this column refers to Vno column of vouchersMain table to maintain referential integrity. Its value cannot be null and therefore its ***Allow Zero Length*** property must be set to No with ***Required*** property being ***Yes***. Since there can be more than one debit/credit Entry against each of the credit/debit entry of ***VoucherMain*** table, the values stored in this column cannot be unique and therefore this column alone cannot be a primary key field.
- *Sno* : This column has been included in the table structure to store serial numbers 1,2,3... to correspond to the serial number of debit/credit entries being referred to in the grid of an accounting Voucher: Debit or Credit. Duplicate values will occur in this field also because the serial numbers of entries across the vouchers are bound to be the same. However, both the columns: vno and Sno together provide a unique value because for every voucher the entries are serially numbered and therefore unique. Both the columns together need be declared as primary key of this table.
- *Code* : This column is meant to store the account codes, which in the context of debit voucher are debit accounts and in the context of credit voucher are credit accounts. This column is also the foreign key column because it references the primary key column of ***Accounts*** table. Its data type and properties must be the same as that of Code column of Accounts table, except that its ***Indexed*** property must be set to ***Yes*** (Duplicates OK). The domain of its data values gets confined to the Code values stored in Accounts table. The data type and properties of ***amount*** and ***narration*** column continue to be the same as already described and discussed for Vouchers table.

15.3 Vouchers Using Forms

The scope of this section includes the basics of Access for creating a Form in Access; transforming the voucher designs in terms of Access objects and properties; and also the procedure for creating Forms for vouchers.

15.3.1 Access Basics for Creating Forms

A Form in Access may be designed, developed and used for the following purposes :

- Data Entry: Form is used for entering, editing and displaying data.
- Application flow : Form is used for navigating through an application.
- Custom Dialog Box : It can be used for providing messages to the user or getting parameters from the user for executing a parameter-based query.
- Printing information: It can be used for providing hard copies of data entry information.

This is contrary to the belief that Forms in Access can be used only for data entry. The most common use of a Form in Access is to display and edit existing data and also for adding new data records.

15.3.2 Tool Box and Form Controls

A tool box is a collection of visual objects (or controls) that are placed (or embedded) on the Form to provide some meaning or functionality. The Form is designed by placing several such controls, which have their own functionality and properties.

15.3.3 Properties of Controls

Every form control is a complete object with its independent set of properties, which determine the shape, size, behaviour and functionality of the object. The properties of these objects are divided into three categories: Format, Data and Others. All these properties may not apply to all the controls. Some important properties of these objects are as described below :

(a) *Format Properties* : Some of the important properties are as described as under:
   - *Format* : It determines the manner in which the data in the control is displayed. This property is inherited from its underlying data source. It is set and used in three situations : *one* when the property is not set for the underlying field; *second* when the format setting of the underlying field is to be overridden; *third* when a control, which is not bound to any underlying data field, is to be displayed in a particular manner.
   - *Decimal Places* : This property specifies the number of decimal places up to which the control should display a numeric data. It must be used in conjunction with format property to determine the final appearance of numeric data.
   - *Caption* : The caption property applies to label, command button and toggle buttons. This property is used to specify what printed matter will appear on the face of the control. In the context of label control, the printed matter is made to appear using this caption property.

- *Visible* : This property specifies whether the control embedded on the Form should be visible or hidden when the Form is opened. The property can make a control appear conditionally when required.
- *Layout* Properties (*Left, Top, Width, Height*) : These properties are used to set the position and size of the control.
- *Back Colour and Style*: The back colour property specifies background colour, as opposed to text colour, for the control. This property, when set to transparent, shows the form's background colour through the control. The setting is preferred for an Option group.
- *Special effects*: This property provides the three dimensional effect to a control in its appearance. The options for this property are: Flat, Raised, Sunken, Etched, Shadowed and Chiseled. Each of these effects give a different look to the control.
- *Border Properties*(style, colour, and effect): The Border properties are capable of affecting the style, colour and thickness of the Border of a control. The Border style options are Transparent, Solid, Dashes, Dots, etc. The Border colour property specifies the colour of the Border and it is possible to select from a variety of colours. The Border width property can be set to one of several point sizes. When the Border style of a control is set to transparent, its colour and width properties are ignored.
- *Fore Colour*: This property can be used for assigning a colour of choice to the text being formatted.
- *Font Properties* (Name, Size, Weight, Italics, Underline): These properties are meant to control the appearance of text within a control. These are capable of affecting font, its point size, thickness and also whether the text is italicised or underlined.
- *Text Align* : The text-align property affects the manner in which data is aligned within the control. The available options are: General, Left, Centre, Right and Distribute.
- *Margins* (*Top, Left, Right and Bottom*) : These properties determine how far the text appears from top, left, right and bottom of a control. The margin properties are of particular importance while using Text box for memo field.
- *Line Spacing*: It is used to determine the spacing between the lines of a text with multiple lines. This is useful when a text control is used for displaying and storing data pertaining to Memo fields.
- *Display When*: This property is capable of deciding whether to send the data of a control to a Printer or to a Screen. For example, the labels containing instructions can be displayed on the screen but not on the printer.

- *Scroll Bars*: This property is capable of determining whether scroll bars appear when the data in the control does not fit within its size or not. The options are none or vertical. This property is normally set to vertical for text control to interact with data pertaining to Memo field.

(b) *Data Properties*

- *Control Source* : This property specifies the field from a record source that is associated with particular control. By default, it is the record source that underlies the Form being designed.
- *Input Mask* : The input mask property affects the format used for data entry into the control as opposed to its appearance, which is affected by Format and Decimal places property. The input mask of the field underlying the control is automatically inherited by the control. However, the input mask property of control in the Form is used to further restrict what data is entered into the field.
- *Default Value* : This property determines the value assigned to the field while adding a new data record. It is inherited from the underlying field of record source to which the control is bound. The default value, when set for control, has an overriding effect over the default value set at the underlying field level.
- *Validation* (Rule and Text) : The function performed by Validation Rule and Validation Text for controls is the same as it applies to Fields of database tables, except that the validation is performed at Form level in case of control and database level in case of fields. In case of bound controls, the user cannot enter data into the control, if the validation rules for control and the underlying field are in conflict.
- *Enabled and Locked* : This property is meant to determine whether focus is allowed on the control or not. If it is set to ***No***, the control appears dimmed and mouse action cannot be performed on such control. This property is useful for calculated controls meant only for display of data. Locked property determines whether the data in the control can be modified or not. This property, when set to Yes, deprives a user the facility to edit data, though the focus becomes available. The two properties interact with one another resulting in following behaviour of control :

Locked	Enabled	Effect : The control can
Yes	Yes	get focus ; its data can be copied but not modified
No	Yes	get focus and its data can be modified
Yes	No	not get focus
No	No	not get focus; its data is displayed dimmed

(c) *Other Properties*

- *Name* : This property allows the designer to provide a customised name to a control. The names assigned by the designer should be purpose oriented so that the design structure of the Form becomes self-documenting.
- *Status Bar Text* : This control specifies the text message that is displayed in the status bar when the control acquires the focus.
- *Enter Key Behaviour* : This property is meant to determine whether the use of Enter key adds a new line in the current control or results in moving the cursor to next control. Its setting is useful for Text control bound to Memo field.
- *Allow AutoCorrect* : This property, when set to ***Yes***, enables the auto correction feature to correct automatically common spelling errors and types. It is useful while using Text control for Memo field.
- *Vertical*: This property is meant to determine whether the text in a control appears horizontally or vertically. The default setting is ***No*** to mean the horizontal. When set to Yes, the text within the control is rotated at 90 degrees.
- *Default* : This property applies to command button and specifies whether the control is a default control on the form or not.
- *Tab Stop* : This property indicates whether the Tab key can be used to enter a control or not. It is desirable to set this property to ***No*** for those controls whose values are rarely changed.
- *Tab Index* : This property is used to set the tab order for the control. This property helps in setting the tab order manually as opposed to automatic setting at Form level.
- *Short cut Menu* : This control is capable of attaching a specific menu to a control and a bar/window gets displayed when the user right-clicks at the control.
- *Control Tip Text* : This property is meant to enter text that acts as a tool tip for the control. The tool tip appears automatically when the mouse pointer is placed over the control and left there for a moment.
- *Help Context ID* : This property indicates the Help topic attached to a particular control.

15.3.4 Common Controls in MS Access

Access provides for a number of controls and more can be added using the add-in-manager in Tools of menu bar. There are three types of controls: Bound, unbound and calculated. ***Bound*** controls are used to display and modify data stored in a data table of database. These controls automatically appear in the Form specified in its display control property and inherit many of the properties

assigned to the field to which such controls are bound. ***Unbound*** controls display information to the user or get data from user that is not going to be stored in the database. A ***Calculated*** control is a special type of control, which displays the derived results of an expression or query. The expression may consist of ready-to-use functions that are meant to make computations by using input values. Some commonly used functions have been discussed and described in Appendix given at the end of the chapter. Therefore, the data in calculated control cannot be modified because it is derived data or information. The value of these controls changes automatically as and when the data, to which the expression of the control is bound, changes. Some of the common controls important for designing a Form are discussed below :

(a) *Label* : This control is used to write dark prints on the Form such as Transaction Voucher, Voucher No, S.No, Debit, Credit, Amount, Narration, Authorised By, Prepared By on the left hand side and "Choose the Account to Debited" and "Choose the account to be Credited "on the right hand side of Access voucher Form design of which is shown in Fig 15.4**.** The attached labels are automatically appended to the Form when other controls such as Text boxes, List boxes, Combo boxes, etc. are added because every such added control has to be labeled to inform the user as to what data to enter or edit through the control. The default caption of the label is the caption of the field that underlies the control to which it is bound. If the caption property of the field is kept blank, the label caption uses field name as its caption.

(b) *Text Box* : This control is included in a Form to provide a blank area for entering the data with or without default values. Blank space next to Amount label, for example, is a text box control to receive the value of amount of voucher. Text box, when bound to a particular field of the table, retrieves and displays the data stored in field for a particular row and is capable of modifying and adding data to the table. The unbound text box is used to get the data from the user for its subsequent use in report for providing report criteria.

(c) *List Box* : This control is used for allowing a user to make a limited choice from a given set of values. The domain of its values is predefined and therefore limited. List control may be used next to Debit and Credit labels in a simple transaction voucher, so as to locate the accounts to be debited or credited.

(d) *Combo Box* : This control combines the features of a list box and text box by allowing a user to select an item from a list or enter a value using the keyboard.

(e) *Sub-form* : Many Forms are based on more than one table with One-to-Many relationship. The records of such tables can be displayed by creating

form within a form, with tabular presentation of records. The Form within a Form (also referred to as Main Form) is called SubForm. The Main Form and SubForm have parent-child relationship. The Control used for creating such a child Form is called SubForm/SubReport. Data records appearing in a grid can be stored in database by using SubForm Control. The SubForm whenever created is listed as an independent object like main form in Database Window. However, the SubForm Control in main Form has three properties for creating a link :

- *Source Object* : It contains name of the Form that is being displayed in SubForm control.
- *Link Child Fields* : These are the fields from the Child form that link the this form to the Main form. These are also referred to as Foreign key of related table.
- *Link Master Fields* : These are the fields from the Main Form that link the Child form to the Main Form. These are also referred to as Primary key of primary table. Make sure the Control Wizards tool is selected before adding the SubForm/SubReport control to the Main form.

(f) *Option Groups* : Control, when applied to Option button, allows the designer to select a particular option from out of a set of mutually exclusive options. This option is useful in designing a common Voucher Form for Debit and Credit Voucher for compound transactions.

(g) *Command Button* : It is meant to execute a defined action on the Form. Access provides for six categories of command buttons as described below:

- *Record Navigation* : The record navigation set of command buttons are meant to facilitate pointer movement on data records. At a point of time, only one row of a table, called data record, is accessed. To access other rows, there has to be a pointer for causing record movement.
- *Record Operation* : There are several operations on data records. These are meant to facilitate such operations as add new record, delete record, undo record, save record, duplicate and print record.
- *Form Operation* : These operations are meant to be performed on the entire form as an object. These are Open form, Close form, Print form, Refresh form data and so on.
- *Report Operation* : These operations are related to the report object. Once a report is created, further actions, which can be taken on such report are Mail report, Preview report, Print report and Send report to file. Access provides separate command buttons for each of these actions.
- *Application* : There are five command buttons especially designed for possible operations pertaining to other application programs. ***Run*** application is meant to execute any existing program ; and ***Quit***

application is used to stop the execution of a running application; ***Run*** MS Excel command button is used for calling the MS Excel, spread sheet program which is part of MS Office package. Similarly, a command button to run MS Word results in calling the text processing program of MS Office package; ***Run*** Note Pad command button when executed calls the text writing program provided by the Operating System-Windows.

- *Miscellaneous* : This category include four command buttons: Auto dialer; Run query, Run macro, Print table. Auto dialer button in a form when clicked is capable of dialing a telephone number, provided a modem is attached and configured in the computer system. Run query command button is meant to execute an existing query. Run macro command button is used to execute a specified Macro and Print table command button, when clicked is capable of printing contents of a specified data table from among available tables in database.

In the example of Access Voucher Form shown in Fig 15.4, four command buttons have been embedded. First button when clicked adds a Record while a click action on the second button results in undoing the record. The third command button is meant to delete a record and the fourth button when clicked saves the record to back-end database tables while in this case it is Vouchers table as already described.

(h) *Control Wizard* : If the selected controls (such as List box, combo box or SubForm) when added to the Form do not invoke the automated wizard, the control wizard need be selected by click action before selecting the control which is to be embedded on the Form for design purposes.

15.3.5 Creation of Form

Access provides for creation of a Form either by Design or Wizard. This can be achieved by double clicking at the database file. Immediately the Database Window appears, which is vertically divided into two parts: left and right. The left side displays a list of database objects such as Tables, Queries, Forms, Reports, Pages, Macros and Modules. The right hand side of Database Window shows the various objects created under each of the classes of objects. At the top of Database Window and just below the title bar, there is a menu bar, which consists of three named menu items: Open, Design and New, and five Icons: one to delete an object, second and third to toggle between Large and Small (default) Icons and fourth and fifth to toggle between list (default) and details.

Select Forms Object : This can achieved by a click at Forms listed as object-class. By default, two items appear on the right side of window: "Create Form in design view" and "Create Form by using wizard".

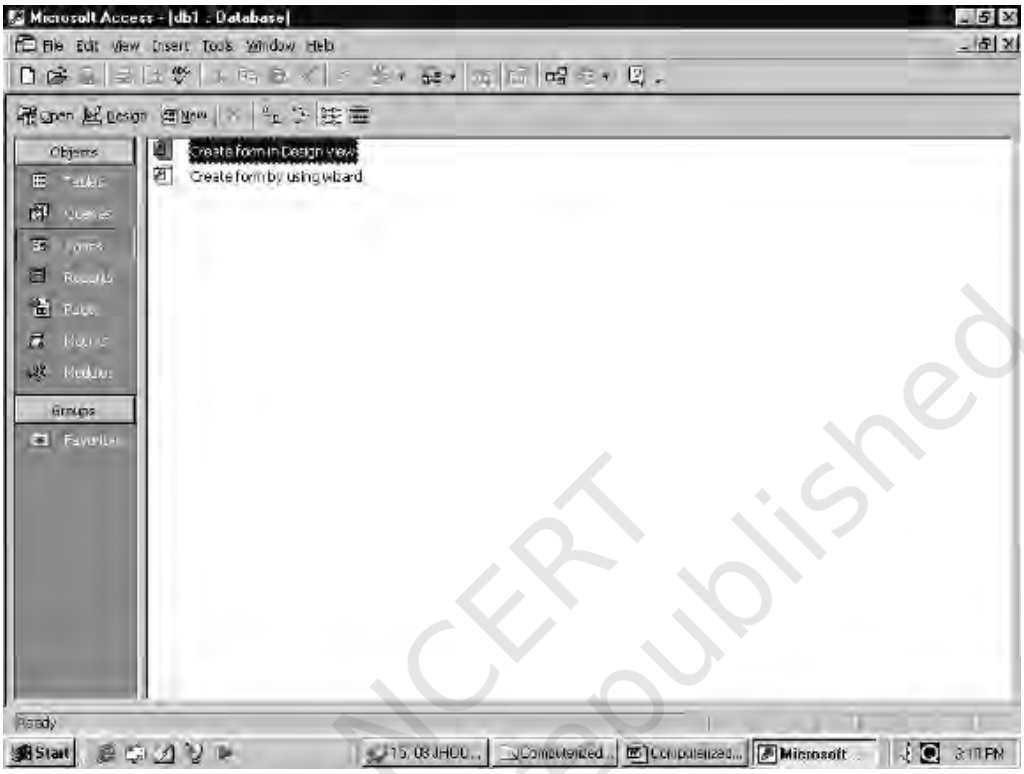


Fig. 15.2 : *Database window showing the methods to create forms*

(a) *Create Form by using wizard* : The following procedure is followed for using the wizard to create a data entry Form :

- Double click at ***Create Form by using wizard***. Immediately there is a window titled, ***Form Wizard*** which allows the designer to choose the data table along with the related available fields to choose from. The designer should choose only those fields, which pertain to the data content of Form being designed. But it must be ensured that every essential field (defined as one with ***Required*** property set to ***Yes*** and ***Allow Zero Length*** property set to ***No***) must be included. In case of voucher, choose all the fields by clicking at **>>** button.
- Click at ***Next*** command button. Form wizard responds by providing six mutually exclusive choices with respect to layout of the Form. One of these choices is exercised by clicking at an option button from a group of six such buttons.

- Click at ***Next*** command button after exercising layout choice. The Form wizard responds by prompting the user to select from a list control one out of the ten options to specify the style of presentation of this Form.
- Click at ***Next*** to move forward. Access responds by asking for the Title of the Form. The designer can provide a useful title, which explains the purpose for which the Form is being created. Further, the designer may specify whether the Form is to be opened for entering data or for modifying the design.
- Finally click at ***Finish*** command button to get the initial design of the Form in run mode, if the option for entering data is exercised. If the option for modify design is exercised, the design of the Form is available along with tool box with various controls to facilitate modification of design.

Modifying Form Design : The Forms created with wizard have limited visual appeal. However, Forms have a design view, just as table do, and Access includes many tools for modifying a Form's design. Some of the common modifications to the Form are listed below :

- Changing Properties of controls
- Re-sizing and moving controls
- Aligning and spacing controls
- Converting (or Morphing) controls
- Conditional formatting of controls
- Re-arranging Tab Order
- Adding New controls
- Deleting existing controls

Each of these modifications has been briefly discussed after describing the procedure for creating a ***Form by Design*** view.

(b) *Create Form by Design view* : Under this method, a data entry Form is created either as a data bound object or as an unbound object. A double click at "Create Form in Design View" provides a ***New Form*** dialog but the Form created in this manner is not bound to any back end database. However, a click at ***New*** to open New Form dialog results in creating the Form, which is bound to database. The use of drop down list in the new Form dialog box to select a table or query serves as the foundation of the Form being created. Fields can be easily added to a Form by using the Field List window, which contains all the fields that are part of the Form's record source. The record source for the Form is the table or query that underlies the Form. Make sure that the Field List window is visible. If it is not, click at the Field List button on the tool bar. Pick up from the field list every field, which is to be displayed in the Form for entering the data. It is important to ensure that every essential field must appear in the Form, if

the Form is being designed to enter records rather than displaying just part of record contents. Select and drag the field from the field list to a place on the Form where it is desired to appear. The location selected becomes the upper left corner of the text box, and the attached label appears to the left of where the text control is dropped. Further, the following steps are taken to develop a data entry Form :

(i) *Click at **New** to open the **New Form** dialog. Two list controls appear in the dialog box* : *one* provides for various options to create a Form such as Design view, Form Wizard, Auto Form; etc. and *another* to "choose the table or query where object's data comes from" (also called record source). From First List control choose Design view(default) by a click.

(ii) Choose a table as the record source because the entire data is stored in the table record by record. Click ***OK*** after the table is selected.

(iii) *Access responds by providing three windows* : *one* for new blank Form, *second* for tool box and *third* for Field list corresponding to the selected record source. The Form object henceforth shall act as a container for other controls to be used in designing Form.

(iv) Select and drag a field from the Field list and place it in the blank Form by drag and drop method. Repeat this process for every field in succession. Alternatively, all the fields can be selected by clicking at every field in the field list while ***Ctrl Key*** is kept ***pressed.*** The selected fields can be dragged and dropped at the Voucher Form.

(v) *Adding a Title* : The Form must be suitably titled for its identity, which should be self-descriptive. To add a title, use tool box by clicking at the label control. While the pointer is moved back into the design area, it changes to a large letter ***A with crosshairs***. Move the pointer into the header area and click where the label is desired to be placed and then type the text of title. Once the text is entered, the focus from the label control can be freed by clicking anywhere in the Form. The label can be reselected by a click, followed by using the formatting tool bar to format the title. Alternatively and in addition to the above, more formatting options can be exercised by right clicking at the label control and clicking at ***Properties*** item of drop down window of right click action.

(vi) *Changing the Properties of Forms and Controls* : Every Access object: Form or Controls is described by its properties. These properties, as already stated above, have been classified into three broad categories: Format, Data and Others. It is not essential to know every available property to work well in designing Forms in Access. But it is always good idea to check up the property values if the object is not behaving the way it is expected to. To view the properties

for an object or control, right click at the control and select the properties. Access responds by providing all the properties listed under category tabs. The property sheet title bar includes names of objects contained in the Form. Once property sheet is opened for one object, it is easy to call for the properties of other objects by selecting the name of object from property sheet title bar. The values of such properties are changed as desired. The Form's property sheet can be opened by double-click at Form selector, which is located at the left most intersection of vertical and horizontal rulers. The property setting on multiple controls can be changed at the same time by selecting multiple objects, in which case only those properties become available for editing which are common to the selected objects. The multiple objects can be selected by keeping the ***Shift Key*** pressed, followed by clicking at desired objects.

(vii) *Moving and Resising controls* : In order to move a control, first select it by a click action, then move the pointer to the edge of the selected control, ensuring that any of the re-sizing handles appearing as bold dot is not pointed at directly. The pointer turns its shape to a small hand. At this stage, hold mouse button pressed and drag the control to its new location. Movement of control beyond the bottom or right edge of the Form, leads to increasing the Form area automatically. Access also allows for combining of select and move step thereby making it easier and more efficient to reposition the control. A control can be re-sized by dragging the re-sizing handles at the corners and sides of the object. A change in the size of text control, however, does not result in changing the size of its underlying field because the size of the field is specified in table's design and can be changed only by modifying the properties of the field in table design.

(viii) *Aligning and Spacing Controls* : Select two or more controls (click at control to be selected by holding the ***Shift Key*** pressed) to be aligned and choose Format-align or right click and choose Align from the shortcut menu to open the list of alignment options. Align-Left leads to aligning the left edges of all the selected controls; Align-Right aligns the right edges of the control. To adjust controls on the same horizontal line, Align-Top or Bottom options can be used. Spacing of controls allows to change (increase or decrease) the relative position of selected controls by one grid point horizontally or vertically. The spacing becomes important when the controls are to be spread out or move closer together for a neater visual layout. Spacing can also be used for ensuring that the controls are evenly spaced.

(ix) *Converting (or Morphing) Controls* : Initially, when a Form is built, it is not always possible to choose the best type of controls to display each field on the Form. One might make a choice for the control only to find out later that it does not suit to the requirements. This is particularly important when the initial design of a Form is created using Form wizard. Access provides for conversion (or morphing) of such control into the desired ones. One of the most common types of morphing is from text to List box or combo box. This is achieved by right click on the text box, followed by choosing ***Change To*** and selecting the type of control to into which text box is to be morphed. Every control cannot be morphed into every other type of control. Text box, for instance, can be converted into a label, list box or combo box. After morphing, a text box to list box, for instance, it is important to modify the control properties such as row source, bound column, column count and column width so that the changed control behaves in a desired manner.

(x) *Conditional formatting of text boxes* : The conditional formatting is displayed in a text control when the value of text control meets a specified criteria or a set of specified criterion. For example, the colour of Amount entered should turn Red when it exceeds a certain limit say Rs. 20,000. In order to create conditional format, right click at text box to be conditionally formatted when in design mode, followed by conditional formatting item of right click window. Access responds by providing conditional formatting window which appears as follows : Conditional formatting window, as shown above, is divided into two parts: the default setting and condition-1. Since the formatting is to occur on the basis of a field value, the criteria list control can be used to select greater than and Rs. 20,000 is entered in the right most box of condtion-1. There are five icons: bold, Italics, underline, Back colour and Fore colour for formatting the data value. As and when the condition is satisfied, the formatting based on the selected icons applies to the data value. If there are multiple conditions for formatting, Add button can be clicked to call for additional formatting conditions. At the most three conditions can be set up for conditional formatting. Click OK to apply the conditions and click Delete to remove the conditional format.

(xi) *Re-arranging the Tab Order* : The tab order of the Form (defined as a sequence of controls to move through when pressing a tab) is assigned while creating a Form. The tab order goes out of sequence when the controls in the Form are re-arranged. An inconsistent tab order leads to an erroneous data entry. To change the tab order, choose ***View-Tab*** order or right click and choose tab order to open

Tab Order dialog box. Clicking ***Auto Order*** generally rearranges the fields in the correct order. It is preferable to try this option first. If the auto order is not correct, the tab order can be set manually by clicking the row selector for a control and then dragging the control up or down into position in the Tab Order.

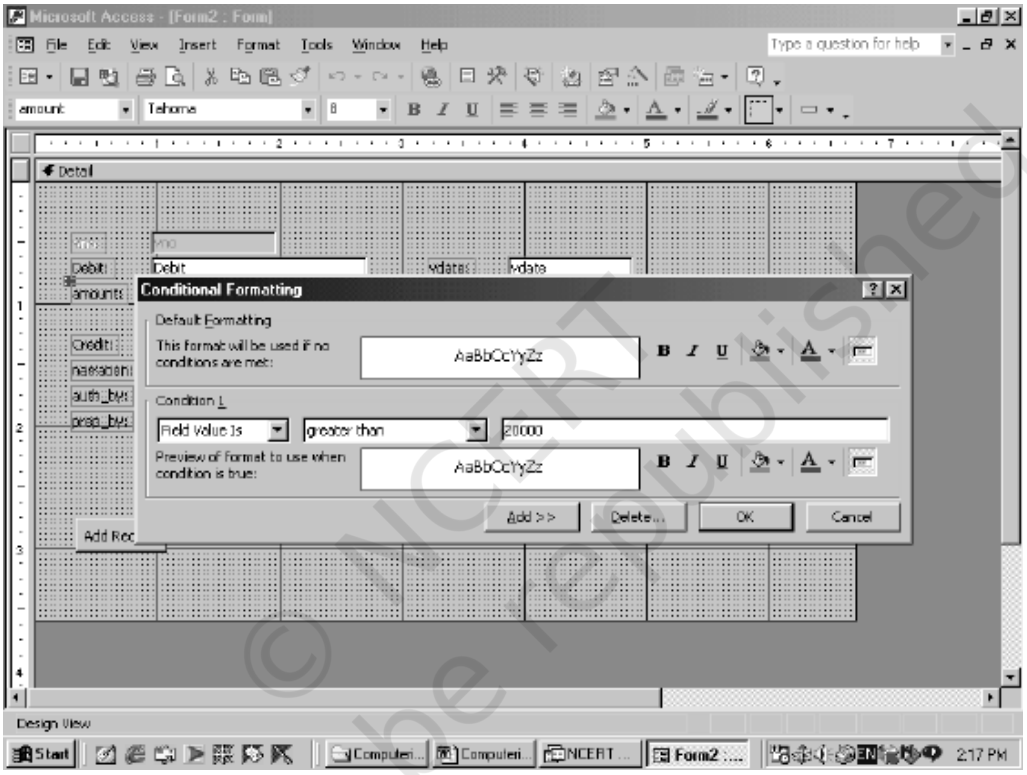


Fig. 15.3 : *Conditional formatting window*

15.3.6 Procedure for Creating Voucher Forms

On the basis of above discussion, the following procedure can be followed to create the different types of vouchers :

(a) Simple Transaction Voucher : The transaction data of simple accounting vouchers is required to be stored in the Vouchers table of a database by using a data entry Form in Access. The format of such a form is shown in figure 15.4.

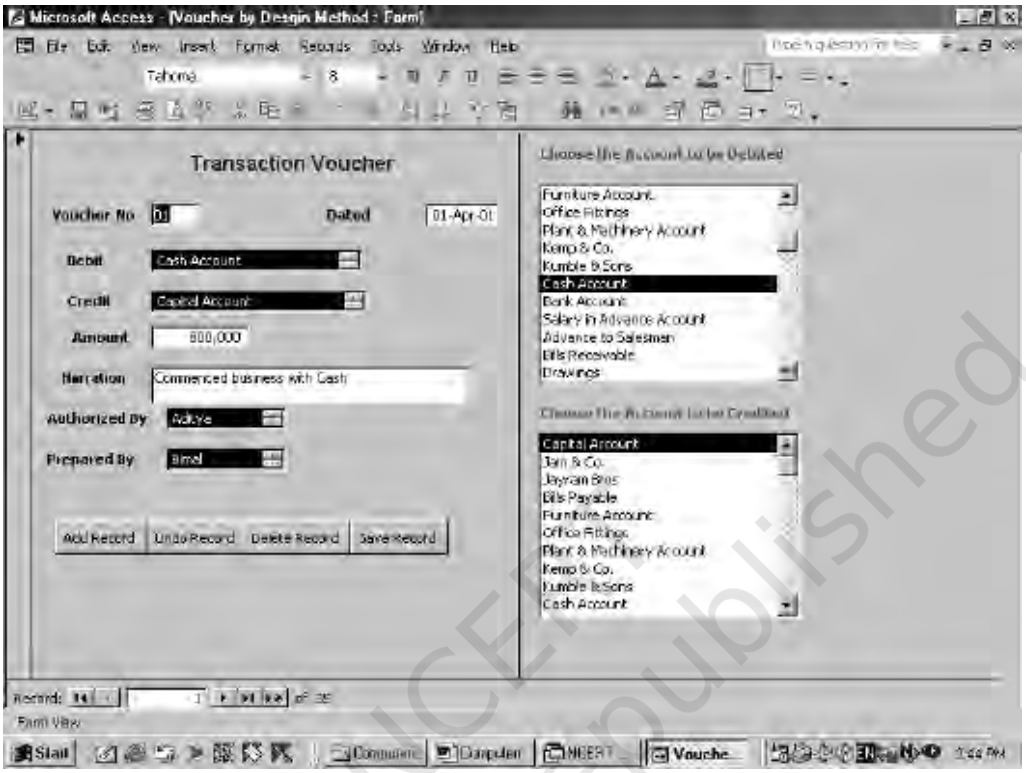


Fig. 15.4 : *Transaction voucher, using database design (model-I)*

The above voucher form uses database design (Model-I) at the backend. A perusal of the this voucher Form reveals that there are two parts: Left and Right separated by a dark vertical line. Left part is dedicated to the data entry of transaction data while the right part has two list controls: one each giving the accounts to be debited and credited. The pre-printed contents of simple transaction voucher appear to the left of above Form as bold dark words. The access resource required to display such pre-printed matter is label control. The data entry spaces against Voucher Number, Dated, Amount and Narration are Text Controls. The list controls have been deployed against Debit Account, Credit Account, Prepared By and Authorised By. The Title of the Voucher Form has been written by using the Label Control. Four operation buttons called Command Buttons control the data entry into the voucher Form. On the Right hand side of above voucher Form, the list controls have been used in expanded Form to choose debit and credit accounts. The resources used in creation of above voucher

Form, therefore, consist of Labels, Texts, List controls and Command Buttons. Once a blank Form is picked up like a container, it is capable of containing these controls including command buttons. The following steps are required for creating the Simple Transaction Voucher as per the Access design given above.

(i) Once the Database Window is opened and Forms object is selected, click at ***New*** item of menu bar. Access responds by displaying a New Form window in which design view option among others appears by default along with a list control to select a table or query which is to act as underlying data source for the voucher being designed. In designing Simple voucher Form, it is fairly clear that the data entered using this voucher Form is to be stored only in the Vouchers table.

(ii) Choose Vouchers table, which has been designed to include the transaction data in each row as a stand-alone record and click OK.

(iii) Access responds by displaying a blank Form object in Form window, along with two other windows: Tool box and Field List of Vouchers table. Expand this Form towards the right and divide it into two parts left and right using line controls of tool box say in the ratio of 3:1.

(iv) Keep the ***Ctrl Key*** pressed and click at every field in Field List Vouchers window. The colour of the list of fields turns blue.

(v) Press at the selected field's area and drag all the fields to left side of blank Form on which data entry contents of voucher are to be located. It may be noted that every data entry control has been assigned to its left an attached label control whose caption is the caption of the fields in Vouchers table.

(vi) Re-position all the controls to their desired location in the left part of the Form and set the font weight property of each to bold. The caption property of each label can be modified to match the pre-printed layout of the voucher.

(vii) Click at ***label*** control in tool box and add it to the centre top of left-hand side of the Form to add the title: ***Transaction Voucher***. Its font size property need be set to 16 with font weight set to ***bold***. Set the fore colour to ***Blue***.

(viii) Paste another text box anywhere in the Form and set its Control Source property as ***=Val(DMax("Vno","Voucher"))+1*** and Visible property to No. Further, Set Default value property of Text Box to the left of label Voucher No. as ***=Val(DMax("Vno","Voucher"))+1***. This ensures that the text control generates a new value one more than the preceding value of last voucher number entered in Vouchers table, as and when a new record is added. As a result, the voucher

number is detected in Vouchers table and incremented by one to auto generate the voucher number sequentially. Further, set the Enabled property to ***No*** so that the auto generated value is not amenable to any changes by the user.

(ix) Set Default value of Text box meant for entering the voucher date as **= *Now*().** This results in giving current RTC date as the default date to voucher as and when a new voucher record is added. Alternatively, click at ***More Control*** button in tool box to select Microsoft Data and Time Picker Control, Version 6. This control provides a user-friendly and interactive method of selecting a date. Set the format property of this control to "3-dpt custom" and custom-Format property to "dd-MMM-yy" by using DT Picker properties dialog. Control source of this control is set to vdate so that the selected date is stored directly into this field.

(x) Set the format property of Text box meant for Amount to ***Standard*** with decimal places to 2. This ensures the appearance of amount up to two decimal places with standard punctuation of numeric values.

(xi) Provide for conditional formatting of amount so that its colour turns red as and when an expense voucher exceeding Rs. 20,000 is not authorised by an employee whose EmpId ='A001'. This can be achieved by a right click at text box for amount to click at conditional formatting. A conditional formatting dialog appears in which condition-1 is to be given as "Field value greater than Rs. 20,000" interactively. Click **Add** button to provide condition-2 as Expression is ***[AuthBy]<>'A001' and [Debit] like '71*'***. Click colour icon to select red colour in condition-2. Click OK to close the conditional formatting dialog.

(xii) Control morphing from Text box to List box is to be applied on four-text control, each one meant to store Debit, Credit, AuthBy and PrepBy. This can be achieved by right click at each of these controls one by one and click at ***Change To*** item of right click window. To begin with, select List Box option for text box next to Label ***Debit***. Height of text box is expanded. Re-size, it to its original shape and right click to select the property window. Select Data properties button to provide the Row Source as ***Account*** table. Click at format properties button to set the column count property to ***2*** and column width property to ***0.5"***. Ensure that the width property of list control for debit is set to a minimum of 1.75" to accommodate the code as well as Name of Account in a row of list control. The process can be repeated for Text boxes next to Credit. Text controls meant for AuthBy and PrepBy can also be morphed into List Box

controls in a similar way, except that the Row Source Property should be Set to Employees table and Column width be set to .33" only because Empl_Id occupies only four text spaces as opposed to Account code, which need six spaces. Width property of these list controls can be suitably adjusted to accommodate both EmpId and Fname of employees authorising or preparing a voucher.

(xiii) Paste List Controls for selecting debit and credit Accounts on the Right Hand Side (RHS) of Voucher Form. Following steps are taken to accomplish this :

- Click at list box control available in tool box and carry the mouse pointer to the right side of the Form. Its shape will turn into a cross with icon of list control. Place it at the top of right part of the Form. Access responds by invoking List Box Wizard, which provides for three options to choose the look up values. By default, the wizard provides for choosing look up values from table or query.
- Click at Next button to get the classified list of tables and queries to choose from. At this stage, choose the Accounts table because domain of accounts to be debited or credited remains confined to accounts available in Accounts table only.
- Click at Next button to get the available fields of Accounts table: Code, Name and Type. Select Code and Name by clicking at > button.
- Click at Next to get a list of accounts with key column hidden. Uncheck the already checked box to display the key column also in list control.
- Click at Next to get an option to select code to store in database.
- Clicking at Next provides two options: One to remember the value for later use and second to store that value in this field. Choose second option and select the debit field to the right of this option as the column against which the key value of accounts from list control is to be stored.
- Repeat the above process to provide for a list control for Account to be Credited.
- Once both the List box controls for debit and credit entries have been pasted, change the caption property of labels attached to such List boxes and write the text "Choose the Account to be Debited" for first list box and "Choose the Account to be Credited" for second. Set the Font weight property to bold, the fore colour to red and green respectively to distinguish between debit and credit list control and re-size the label control by increasing its width to accommodate the text to caption of label. Re-size the

list boxes to adjust their width and height appropriately. This can be achieved by right clicking at each of the controls to get the property windows.

(xiv) Click at Command button in Tool Box and carry the mouse pointer to Area at the bottom of Left most bottom corner of the Voucher Form. Its shape turns into a cross with command button icon. Paste it by horizontal and vertical dragging to give suitable width and height. Immediately, the Command Button wizard is invoked to seek information about category of operation and the action to be performed using this command button. Choose Record operation as category with Add New Record as the action. Click at Next to state whether the caption of the command button is to be a text value or an icon. A click at next after appropriate selection results in giving a suitable object name to the command button. Accept the default value and click at finish. This results in pasting an operational button on the Form with the capability to add a new record.

(xv) Repeat this action to create various other command buttons to match the design of Transaction Voucher Form given above.

(b) Compound Transaction Voucher : The transaction data of Debit or Credit vouchers, which have already been described as compound transaction vouchers, is required to be stored in VouchersMain and VouchersDetails tables of database. Its design, when transformed in Access Form layout, is expected to appear in the following format :

A perusal of the above Access Form for Credit Voucher reveals that there are four labels: Voucher No, Date, Prepared By and Authorised By in Dark bold letters. These labels are meant to define the pre-printed content of the voucher as per design. Next to first two labels: Voucher No. and Date are text boxes displaying their respective data contents. To the right of labels Authorised By and Prepared By are List Box controls to get and display the first name of employees. Text Box displays the Title of the Voucher Form ***Credit Voucher*** as calculated control because the same voucher design is used for Debit vouchers also. Just below this dynamic title is the Option group control whereby the user can make a mutually exclusive choice for Debit or Credit Voucher. The title of Entries Grid and Text box to the left of a list control are used to select an account to be debited or credited (the complementing account) against the accounts being mentioned in the Entries Grid are also calculated text controls. The calculated text controls acquire the text value to display on the basis of what is selected in the Option groups. Next to calculated text box control is a label to print an instruction for refreshing the display in grid. The grid consists of five columns: S.No, Code, Name of Account, Amount and Narration. The grid appears in the voucher by using SubForm control. Besides this, there are five command buttons, each dedicated to Add

Record, Undo Record, Delete Record, Save Record and Close. These command buttons operate on the data entry Form.

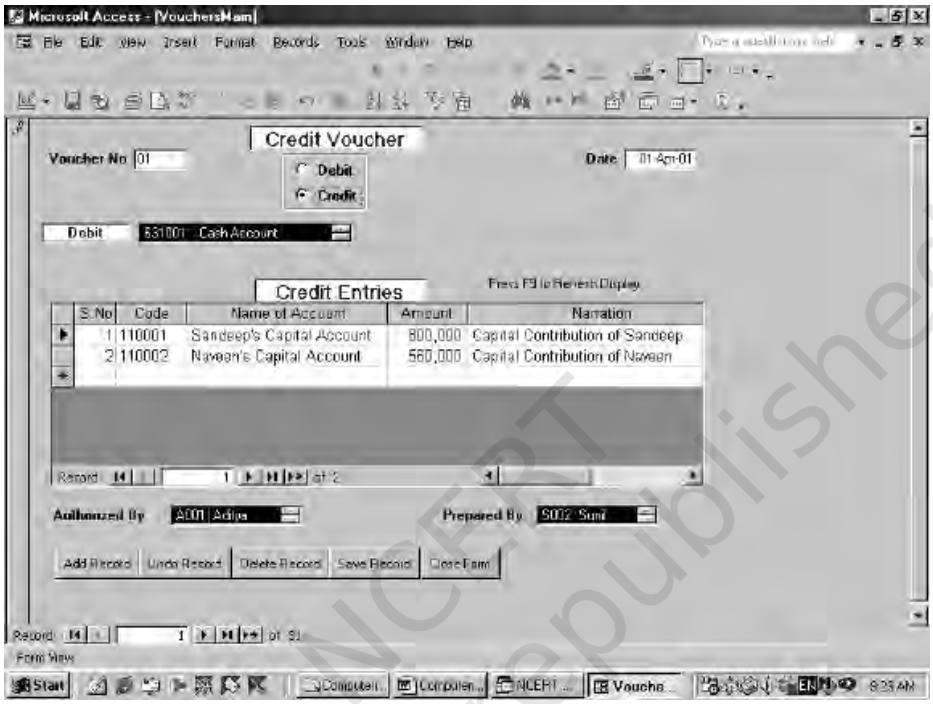


Fig. 15.5 : *Credit voucher created as a form in access*

To create this voucher Form, following steps are taken using design view :

(i) Create a blank form in design view and ensure that its underlying data source is selected as ***VouchersMain*** table and Field List window along with tool box is also displayed. As already discussed in Section I of this chapter, the Compound Voucher Form requires another related data table, ***VouchersDetail,*** for storing the data contents of grid.

(ii) Keep the ***Ctrl key*** pressed and click at Vno,Vdate, AuthBy and PrepBy fields in Field List window.

(iii) Press at any of the selected field's area, drag and drop it to blank Form. It can be observed that all the selected fields are also dragged and dropped along with this field.

(iv) Re-position all the controls to their desired location in the Form and set the font weight property of each to bold. The caption property of each label can be modified to match the pre-printed layout of the voucher.

(v) Paste Option group control just below the form space meant for dynamic title. Access responds by prompting the user to enter the label names for each option. Enter two options by writing Debit and Credit in different rows. This must be followed by a click at ***Next*** button.

(vi) Option group wizard responds by prompting the designer to enter the default option.

(vii) Select Debit so that by default, the compound voucher is a Debit Voucher. Click at ***Next*** button. Access responds by prompting the designer to enter the data values corresponding to each of the option labels. Enter against Debit and Credit **0** and **1** respectively. Click ***Next*** button.

(viii) Access Responds by requiring the user to either opt for ***Save the value for Later Use*** *or* ***Save the value in this Field***. Choose the second option for ***Save the value*** and select Vno as Type field. Click at ***Next*** button.

(ix) Access responds by asking the designer to choose the appropriate control type. Choose Option buttons, along with any of the styles given below in wizard dialog. Click ***Finish*** button. Access assigns a default label to the Option group. Select the label by right clicks and remove it by clicking at ***Cut***.

(x) Click at text control in tool box and add it to the centre top of the Form to provide a dynamic text for the title: ***Debit or Credit Voucher***. The attached label control is removed by a right click on this label followed by a click on ***Cut***. The font size property of Text need be set to 16 with font weight set to bold. Set the fore colour to Blue. Set its Control Source property as ***= IIF([Type] = 0,"Debit","Credit") & " " & "Voucher"*** Re-size the width of this text control so that it can accommodate and display the dynamic title of voucher. By entering the above formulae in Control Source property, text control for title becomes dynamic. Whenever, the Type field is assigned 0 value, a text control for title displays Debit Voucher and when the value of Type is set to 1, the Credit Voucher is displayed by the this Text control. The title of simple Transaction Voucher is static. Therefore, a label control has been used for this purpose. Further, set the ***Enabled*** property to ***No*** so that the displayed text is not amendable to any changes by the user. This applies to other similar controls meant for dynamic texts in this Voucher Form.

(xi) Paste another text box anywhere in the Form and set its Control Source Property as ***= Val(DMax("Vno","Voucher")) + 1*** and ***Visible*** property to ***No***. Further, Set Default value property of Text box to the left of label Voucher No. as ***=Val(DMax("Vno","Voucher"))+1***. This ensures that the text control generates a new value one more

than the preceding value of the last voucher number entered in vouchers table as and when a new record is added. As a result, the voucher number is detected in Voucher table and incremented by one to auto generate the voucher number sequentially. Set its ***Enabled*** property to ***No*** for reasons already explained earlier.

(xii) Set Default value of Text box meant for entering the voucher date as **= *Now*().** This results in displaying RTC date as the default date to voucher as and when a new voucher record is added.

(xiii) Paste another text control below, the label ***Voucher No*:** to indicate Debit in case of Credit voucher and Credit in case of Debit Voucher. Remove its attached label and set its Font size and font weight property appropriately. However, its Control Source property is set as **= *IIF([Type] = 0,"Credit","Debit")*** so that this text box displays the desired text as stated above. Pick up a List control from tool box and place it next to this calculated text control to choose the account. Immediately, the List control wizard gets activated and displayed. Complete the list control creation process as already discussed while designing the simple transaction form. Ensure that its Control source property is assigned the Field name ***AccCode***; Row Source to ***Accounts***; Column Count set to **2**; Bound Column set to **1** and Column width to **0.5"**. Re-size the control for proper display.

Creating Grid for Debit/Credit Entries : The grid for entries is created by using SubForm Control. Following steps are taken to create SubForm to be linked to Main Voucher Form :

(i) Pick and paste SubForm control for creating a grid to accommodate the Debit/Credit Entries. SubForm wizard gets activated and displayed. Choose existing Tables/Queries, followed by click at ***Next*** button. Subform wizard displays a dialog to giving fields classified by their respective tables. Choose Sno, Code from ***VouchersDetail*** table; Name from Accounts table; again Amount, narration and Vno from ***VouchersDetail*** table. Click ***Next*** button.

(ii) Choose "Show VoucherDetail for each record in VouchersMain using Vno" and Click ***Next*** and provide the name for subform object as "VouchersDetail SubForm" Click at ***Finish***. The SubForm stands created to accommodate the data contents in voucher grid. The attached label of SubForm is removed to pave the way for creating dynamic title. This is achieved by adding another text control (remove the attached label control) at the top of the SubForm in the same manner as applies to the title of voucher, except that the Control Source property is set to **= *IIF([Type] = 0,"Debit","Credit") & " " & "Entries*"**. This calculated control is capable of showing the title of the grid as Debit Entries or Credit Entries, depending on choice of Option button at run time.

The Voucher number column in grid can be hidden by merging its right most vertical line with vertical line separating narration and voucher number column by drag and drop method.

(iii) Set the format property of Text box meant for Amount to ***Standard*** with decimal places to **2**. This ensures the appearance of amount up to two decimal places with standard punctuation of numeric values.

(iv) Provide conditional formatting of amount so that its colour turns ino red as and when an expense voucher exceeding Rs. 20,000 is not authorised by an employee whose EmpId ='A001'. This is achieved by a right click at text box for ***amount*** to get short-cut window so that conditional formatting item is selected. A conditional formatting dialog appears in which condition-1 is to be given as "Field value greater than Rs. 20,000" interactively. Click ***Add*** button to provide condition-2 as Expression is ***[AuthBy]<>'A001' and [Debit] like '71*'***. Click colour icon to select Red colour in condition-2. Click ***OK*** to close the conditional formatting dialog.

(v) Text control for entering Code in SubForm can be morphed to List control in the same manner as already explained for Debit/Credit Account in simple Transaction Voucher except that the Control source property is assigned the Field name Code of ***VouchersDetail*** Table.

(vi) Control morphing from Text box to List box is also to be applied on text controls meant to store the data values for AuthBy and PrepBy respectively. This can be achieved in the manner as already described in the context of designing a simple Transaction Voucher.

(vii) Paste a label control to the top right of SubForm for displaying the instruction "Press F9 to Refresh Display".

(viii) Command button at the bottom of Debit/Credit Voucher Form can be added in the same manner as described above in the context of Simple Transaction Form. An additional Command button with Caption ***Close Form*** can be added by choosing Form Operation as category with ***Close Form*** as the action.

While operating on the above form in run mode, it must be ensured by the user that the entries in the grid are made only after saving the data contents of voucher outside the grid. This is because a data record for contents outside the grid belongs to VouchersMain table. Such record in primary table must exist before any data record is entered in grid to be finally stored in ***VouchersDetail*** table.

15.4 Information Using Queries

Accounting information that is presented in an accounting report is generated by creating and executing various queries using DBMS. The basics of creating such queries in MS Access have been described below along with their usage in the context of Model-1.

15.4.1 Basics of Creating Queries in Access

Recall that one of the great advantages of relational databases is that the fragmented data is stored in different data tables so that there is no or minimum redundancy. But a complete view of data stored across various tables is achieved only by executing queries based on SQL. A query is capable of displaying records containing fields from across a number of data tables.

15.4.2 Types of Queries

There are several types of queries in Access that are used to generate information. Such queries are called select queries because they are used to "select" records with a given set of fields: actual and computed and also for a given criteria. There are three important query types that are required for generating the accounting reports. These queries have been discussed as below:

(a) *Simple Query* : A select query is a simple query if it does not involve use of any query function to produce a summary of data. The criteria, if any, used in such a query is based on some constant value or values, forming an integral part of the query. For example, a query, to find date and amount of transactions records in which an account, identified by code = '711001' is debited, is a simple query and is executed, using database design of Model-I by the following SQL statement :

```
SELECT vDate, Amount
FROM Vouchers
WHERE Debit = '711001'
```

In the above SQL statement, the SELECT statement is meant to specify the fields to be selected, FROM clause specifies the source of data and WHERE clause filters the records matching the condition that Debit field has code = '711001'

(b) *Parameter Queries* : A parameter query prompts the user to enter parameters, or criteria through an input box, for selecting a set of records. A parameter query is useful when there is a need to repeat the same query with different criteria. The criteria, this means, is not constant as in the case of the simple query. While extracting the transactions to prepare ledger accounts, the same set of queries need be executed for different account codes. Consider the following SQL statement :

```
PARAMETERS AccountName Text (255)
SELECT Name
FROM Accounts
WHERE Code = AccountNo
```

In the above query, the PARAMETERS clause is meant to declare the variable AccountNo. This SQL statement, when executed, prompts the user to provide the value of AccountNo.

(c) *Summary Queries* : A summary query, as opposed to a simple query, is used to extract aggregate of data items for a group of records rather than a detailed set of records. This query type is of particular importance in accounting because the accounting reports are based on summarisation of transaction data. Consider the following SQL statement :

```
SELECT Code, Name, Sum(Amount)
From Vouchers INNER JOIN Accounts
ON (Accounts.Code=Vouchers.Debit)
GROUP BY Code, Name
```

In the above query, the ***Vouchers*** table has been joined with ***Accounts*** table on the basis of ***Code*** field of Accounts and ***Debit*** field of Vouchers. The resultant record set has been grouped on the basis of Code and name of accounts. Accordingly, the sum of amount for each group (or set of records) has been ascertained and displayed. Finding the sum is the process of summarisation.

15.4.3 Adding Computed fields

The computed fields, representing secondary data, do not form part of data stored in tables because such data items unnecessarily increase the size of database. The secondary data items can always be generated on the basis of primary (or stored) data. In order to find values of such secondary data items, the query is based on computed fields. The computed fields provide up-to-date calculated results because they rely upon updated stored data values. For example, a data table, named *Sales*, which includes ItemCode, Quantity, Price, Dated and CustId, is maintained in a database to store sales transactions. In order to get list of sales transactions along with total sales relating to CustId='A051', the following simple query is executed by including Sales as computed field :

```
SELECT Dated, ItemCode, Quantity*Price AS Sales
FROM Sales WHERE CustId= 'A051';
```

In the above query the expression Quantity*Price has been given the name Sales by using ***AS*** clause.

15.4.4 Using Functions in Queries

A function in the Access environment is named and followed by parenthesis ( ). The function receives some inputs as its arguments and returns a value (also called its output). These functions also form a part of the expression for a computed field. Some commonly used functions have been described and discussed in Appendix given at the end of the chapter.

15.4.5 Methods of Creating Query

There are three ways in which any of the above queries can be created in Access. These methods are Wizard, Design and SQL View. A brief description of each is given below :

(a) *Wizard Method* : In order to create a query using Wizard, the following steps are required :

(i) Select Queries from Objects list given in LHS (Left Hand Side) of Database window.

(ii) Double click at ***Create Query by Using Wizard*** given on the RHS (Right Hand Side). Immediately, there is a window titled 'Simple Query Wizard' (Shown in figure: 14.6) that prompts the user to select a field from a table or an existing query that is to be included in the query being created. Many such fields may be selected according to the information requirement of the query. The tables (or queries)

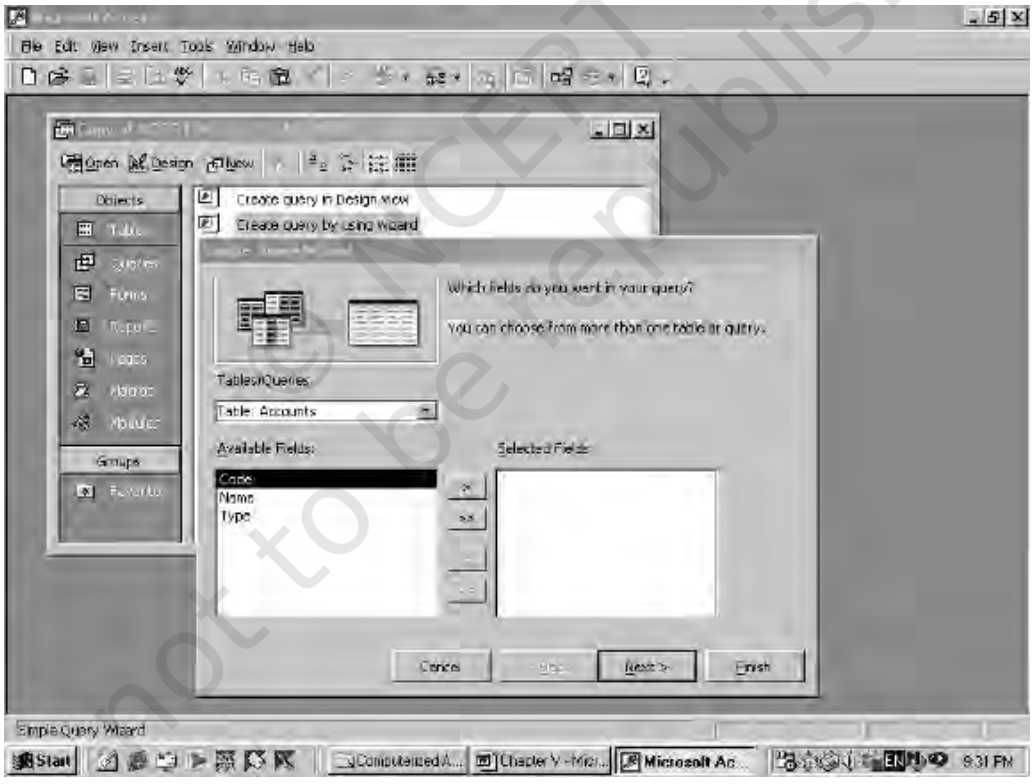


Fig. 15.6 : *Window to display simple query wizard*

being chosen represent the data source of the query being created. The fields being selected imply the data items to be displayed by the query. Use arrow buttons or double click at the list of fields on LHS of this window to select fields.

(iii) Click at ***Next*** after the desired fields have been selected. If the selected fields include a number or currency field, the designer is prompted to choose an option button to specify whether the query to be created is a summary or detail query.

- If detail option is chosen, the execution of query results in displaying records from data source.
- If summary option is selected, the user is prompted to indicate the type of summarisation required: Sum, Average, Minimum and Maximum with respect to the field of summarisation. Clicking at check boxes against different types of summarisations specifies this. Click ***OK.***

(iv) Click at ***Next*** and specify the name of the query being created **%** ***Finish*** to save and execute the query. The results of the query are displayed in datasheet view.

(b) *Design Method* : In order to create a query by design method, the following steps are required :

(i) Select Queries from Objects list given in LHS of database window. Double click at ***Create Query by Using Design View*** given on the RHS.

(ii) Access responds by displaying a ***Select Query*** and ***Show Tables*** Window. The Select query window is vertically divided into two panes: upper pane and lower pane, as shown in Figure: 15.7. The upper pane is meant to display data sources (Tables or Existing Queries) and the lower pane, which also called Query By Example (QBE) grid, has one column each for field to be included in query being created. The row of this grid shows field name, table (or query), sort order, whether the selected field is shown in the query results or not and also the criteria that have been applied to the field or fields to restrict the query results. The Show Table Window is meant to add tables, queries or both to the upper pane of Select Query Window. If closed, the Show Table Window can be recalled by a right click at upper pane **%** show table.

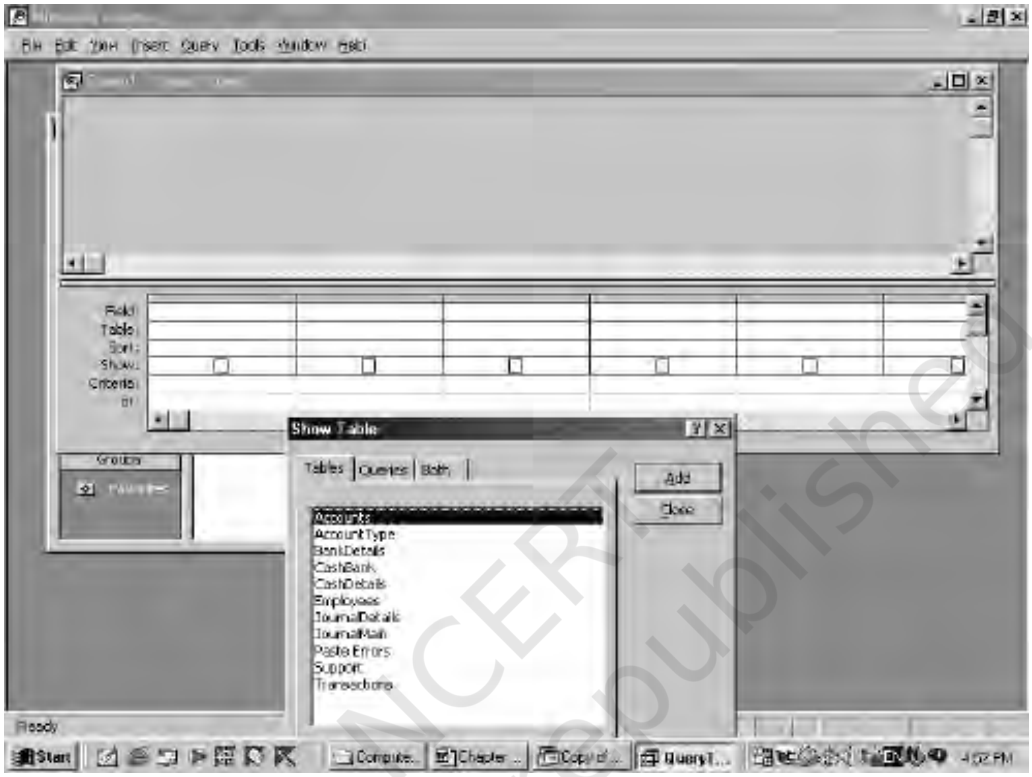


Fig. 15.7 : Select query and show tables windows

(iii) Click at View item of Menu bar % Total and then % Table Names.

(iv) Click at *field* row of first column of QBE grid to select the fields to be included in the query. The process is repeated for second and subsequent columns of grid to include more fields in the query. This process of selection constitutes the data items to be displayed by SELECT clause of SQL statement.

(v) The name of table or query is displayed, in accordance with selection of fields. Such tables or queries constitute the data sources shown after FROM clause of SQL statement. However, the initial selection of a table/query in the second row of QBE grid restricts the choice of fields to the selected table/query only.

(vi) Click at row of grid to specify the Group by clause and aggregate functions so that summary a query is created.

(vii) Click at row of grid to specify the sort order (Ascending or descending) on field(s). The selected fields for sort order are shown after ORDER BY clause of SQL statement in which ascending order is the choice by default.

(viii) Click at row to check for the selected field to be displayed in the query result. The field(s) may be selected only for the purpose of specifying the sort order or criteria.

- Click at row of the grid to specify the criteria to limit the records to be displayed by the query being created. The specified criteria result in a conditional expression, which is shown after the WHERE clause of SQL statement.
- Click ***File % Save*** (or Press Ctrl+S) to save a query. A dialog box prompts the user to specify the name of the query being created. By default a generic name appears which can be accepted or rewritten with a desired name.

(c) *SQL View Method* : A query may be directly specified in Select Query Pane by a right click at table pane % SQL view. The upper and lower panes of selected query window are substituted by a pane to specify the SQL statement that is written by using keyboard. The desired SQL statement is directly okeyed in on this pane and saved in the same manner as described for design method. While forming the SQL statement, the following clauses are normally used for generating information (or Select) queries :

(i) *SELECT* : This clause is used to specify the fields to display data or information. Consider the following SQL statement segment :

SELECT Code, Name, Amount

The fields Code, Name and Amount after SELECT clause indicate the data items to be displayed by the query statement.

(ii) *FROM* : This clause is meant to indicate the source of data in terms of tables or queries or a combination of both. Two tables are joined by specifying a JOIN clause based on a condition of Join. There can be three types of Join: Inner, Left and right.

(iii) *INNER* : This Join clause is meant to display only exactly matching records between two data sources. Consider the following SQL statement segment:

FROM Accounts **INNER JOIN** AccountType
ON (CatId=Type)

In the above statement, only those records of Accounts and AccountType table constitute the source of query data, which match exactly on CatId = Type.

(iv) *LEFT* : With this Join, all the records in the primary table in the relationship are displayed irrespective whether there are matching records in the related table or not. Consider the following SQL statement segment :

FROM Accounts **LEFT JOIN** AccountType
ON (CatId=Type)

In the above statement, all records of Accounts along with matching records of AccountType table constitute the source of query data, The matching condition is CatId = Type.

(v) *RIGHT* : With this Join, all the records of related table in the relationship are displayed irrespective whether there are matching records in the primary table or not. Consider the following SQL statement segment

FROM Accounts **RIGHT JOIN** AccountType
ON (CatId=Type)

In the above statement, all records of AccountType along with matching records of Accounts table constitute the source of query data. The matching condition is CatId=Type.

(iv) *WHERE* : This clause in SQL statement is used to provide the condition to restrict the records to be returned by query. The resultant records of query must satisfy the condition which is specified after WHERE clause. This is meant to filter records returned by the query.

(v) *ORDER BY* : This clause is meant to specify the order in which the resultant records of query are required to appear. The basis of ordering is determined by the list of fields specified after the order by clause. Consider the following SQL statement segment :

ORDER BY Type, Code

The above statement in the context of Accounts table implies that the resultant record set is ordered by the **Type** field of Accounts and within Type, by **Code** field of Accounts.

(vi) *GROUP BY* : The group by clause is used in the SQL statement to enable grouping of records for creating summary query. The fields after GROUP BY clause constitute the basis of grouping for which summary results are obtained. Consider the following SQL statement:

SELECT Debit, **Sum(**Amount**)**
FROM Vouchers
GROUP BY Debit

In the above SQL statement, the GROUP BY clause uses Debit account codes as the basis for computing the sum of amount of voucher. The total amount, by which every transacted account has been debited, is given by this SQL statement In this case, sum of amount is found for each group of records formed using GROUP BY clause.

15.5 Generating Accounting Reports

An Accounting system without reporting capability is incomplete as reporting is one of the main purposes for which an accounting system is designed and operated upon. The output of accounting system takes the form of accounting reports. Access offers a great flexibility in designing and generating customised reports.

15.5.1 Accounting Reports

Every report consists of 'information', which is different from 'data'. Data processing leads to data transformation and when this processing is in accordance with decision usefulness, it is called information. Information generation is the process of compiling, arranging, formatting and presenting information to the users. A report is prepared with a definite objective. Every report is collection of related information for a particular need and purpose and must meet the twin objectives of reporting : *one* to reduce the level of uncertainty that is faced by a decision-maker; *second* to influence the behaviour (or positive actions) of the decision-maker. Accordingly, accounting information, generated by processing accounting data is gathered to generate an accounting report. An accounting report, therefore, is the physical form of accounting information. Useful accounting information, regardless of its physical form, must have five characteristics: relevance, timeliness, accuracy, completeness and summarisation. An accounting report, in order to be useful, must display information content in such a manner as to give confidence to the user, influence his behaviour and prompt him to take positive actions. Reports, which do not meet the above stated objectives, lack or do not have sufficient information content, have no value. There are two broad classes of accounting reports: Programmed and Casual (also called Adhoc or Pass through).

(a) *Programmed Reports* : These reports contain information useful for decision-making situations that the users have anticipated to occur. There are two types of reports within this report type: Scheduled and On demand.
   - *Scheduled Reports* : The reports, which are produced according to a given time frame, are called scheduled reports. The time frame may be daily, weekly, monthly, quarterly or yearly. Some examples of scheduled

reports are: Trial Balance, Ledger, Statement of Cash Transactions (Cash Book), Statement of Ageing Accounts, Closing Stock Report, Profit and Loss Account and Balance Sheet, etc.

- *On Demand Reports* : The reports, which are generated only on the triggering of some event, are called On demand reports. Some examples of On demand reports are a Customer's Statement of Account, Inventory Re-order Report, Stock in hand Report for a Selected Group of items, etc.
- *Casual Reports* : There are reports, the need for which is not anticipated, the information content of which may be useful but casually required. These are adhoc reports and are generated casually by executing some simple queries without requiring much of professional assistance. As opposed to programmed reports, casual reports are generated as and when required.

15.5.2 Process of Creating Reports

The process of generating accounting reports in Access involves three steps: designing the report, identifying the accounting information queries, and finally creating an accounting report by using such queries.

(i) *Designing the Report* : Every report is expected to meet certain objectives of reporting for which it is designed and developed. It should not be too big so as not to be read at all or too small so as to conceal certain vital information of importance that is expected to facilitate decision-making. Objective-oriented reporting means designing the report in such a manner as to meet the pre-conceived objectives in view.

(ii) *Identifying Accounting Information Queries* : A number of SQL statements are written in such a manner that each successive SQL relies on the results of the preceding SQL statement and refines its results by using fresh data (or information) from existing data tables (or queries).

(iii) *Using the Record set of Final SQL :* The record set of final SQL that relies upon preceding SQL statement, is collection of report-oriented information. This record set need be embedded in the report being produced.

15.5.3 Basics of Designing a Report in Access

A report, in Access, is a static presentation of stored or transformed data in an organised manner. Access saves the design of the report, which consists of information structure along with various controls to display information content and its record source. When a saved report is opened, the information content is retrieved from the tables and displayed according to the design. As a result, a saved report design, when opened, displays the information content

according to the current state of data. There are two types of formats of presenting information through a report: Columnar and Tabular.

- *Columnar Report Format* : A columnar format displays the caption of each field on a separate line in a single column down the page. The corresponding information contents of the fields are shown in another column next to their respective fields. If the caption property of a field is kept blank, the name of the field is used as its caption. This implies that there are two columns in this format: *one* for displaying the fields and *another* for showing the corresponding information content. A record set that consists of nine fields, when presented in such a format, requires nine lines of report. In columnar format, the total number of lines to be printed equals the number of fields multiplied by the number of record sets to be displayed.
- *Tabular Report Format* : A tabular format displays the caption of fields on the same line so that their respective information contents appear in the next line. The number of columns in tabular report is exactly equal to the number of fields to be displayed. It implies that the above mentioned record set, when presented in tabular format, requires one line for captions of fields and another line for information content. In tabular format, the total number of lines to be printed equals the number of record sets to be displayed plus one for captions of fields to constitute column headings.

15.5.4 Structure of Report in Access

A report in Access is designed using seven sections which taken together constitutes the structure of report design. It is not necessary that every report designed in Access must have all the sections that have been described below:

- *Report Header* : Report header appears at the top of the report and may include title and other relevant information pertaining to the report.
- *Page Header* : Page header appears at the top of every page of the report. It may include a uniform title to indicate that the page belongs to a particular report.
- *Group Header* : The group header and footer are available in a report only if the sort order and grouping levels are also defined on the basis of a field of data source. This is because Group Header and Footers are properties of the field that are used for defining the sort order. Depending on grouping level, the group header appears at the top of each report group. A set of report pages constitutes a report group. Each group level of report contains a separate group header.
- *Details* : The details section, which is also called the main body of a report, contains data from tables or queries that provide the record source to a report. This section is most important as it consists of the main information content of a report.

- *Group Footer* : The group footer appears at the bottom of each grouping level and may contain summaries or sub-totals for the grouped data.
- *Page Footer* : The page footer appears at the bottom of each page of the report and is meant to include page numbers, date and time of report generation.
- *Report Footer* : The report footer appears once on the last page of the report to include summaries or totals for all data of the report.

It is not necessary to incorporate each and every section or component of report structure. Those report structure components, which are not required in a specific report being designed, are suppressed. To achieve this suppression, open the ***View*** Menu to hide or display the Report Header/Footer, Report Page Header/Footer. The size of every section or report structure component is increased or decreased by dragging section bars up or down using a mouse.

15.5.5 Methods of Creating a Report

There are three ways in which a report can be created in Access. A brief description of each method is given below:

(a) *Auto Report*: This is the easiest method of creating a report both with columnar and tabular formats. To begin with formulate, create and save a query, which is capable of providing a record set as the information source of report. Alternatively, the information content must be available in a single table of the database. If the information is generated by relying upon more than one table, query is the option to be exercise. After the information source becomes available in the database, the following procedure is adopted to create Auto Reports.

(i) Select Reports from objects list given in LHS of Database window and click at ***New*** object button of tool bar. Access responds by displaying the following ***New Report*** *Window*.

(ii) Choose ***AutoReport: Columnar*** or ***AutoReport: Tabular***, followed by selecting the information source query or table.

(iii) Click ***OK*** to generate the report. Access responds by creating and displaying the report in printpreview mode.

(iv) To print the report, click at the print icon on tool bar.

(v) To save the report design as object, close the print preview window, and provide a suitable name.

Auto Reports are easy and fast to create. But these reports are less attractive. To prepare more professional report, report wizard is used.

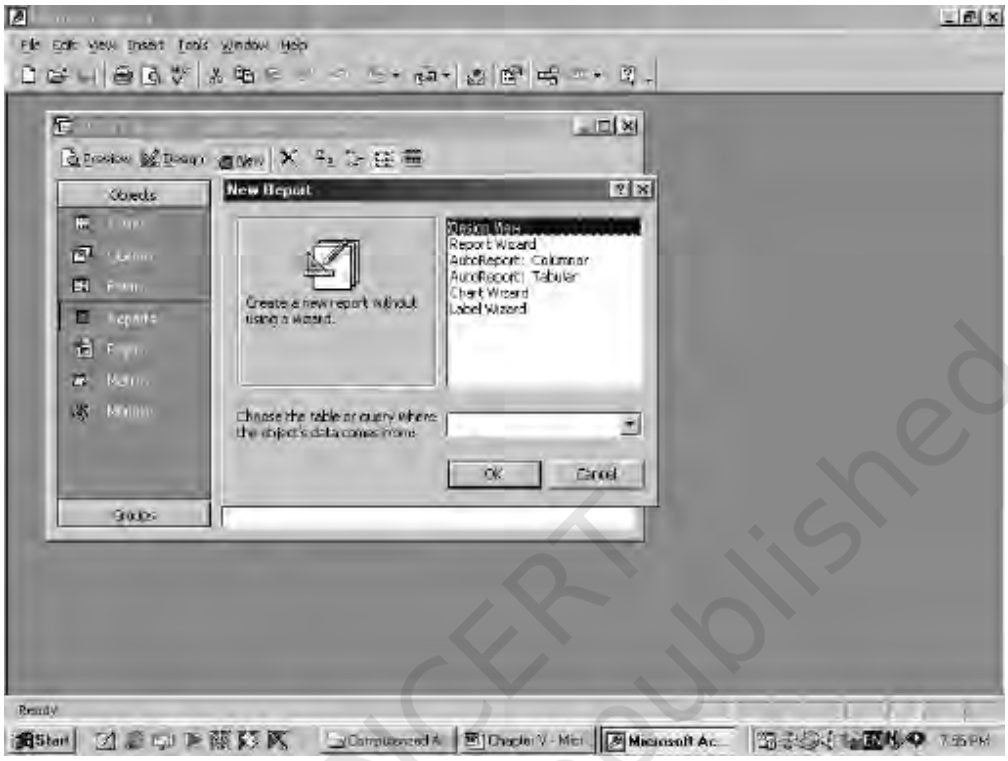


Fig. 15.8 : *New report window to choose methods of report design*

(b) *Wizard* : The Report wizard allows a designer to choose the fields from multiple tables along with specification for grouping, sorting and formatting of information content in report. This obviates the limitation of Auto Reports. In order to create reports by wizard, following steps are required.

(i) After selecting Reports object, double click at ***Create Report by Using Wizard***. Access responds by displaying Report Wizard window similar to the one displayed for query wizard (See Fig 14.10).

(ii) Choose the table or query that includes information content of report, from Tables/Queries drop-down list on LHS.

(iii) Use arrow buttons to select fields to provide the information source to report. Single right arrow button is used to select one field and double arrow button to select all fields. Alternatively, double click at the fields to be selected in the same order in which they are required to be displayed in the report.

(iv) Another table or query can be chosen to select more fields for a report to provide a definite relationship between the tables is defined. Click ***Next*** when selection process of data source is complete.

(v) Access responds by prompting the designer to add any grouping level(s) for displaying the information content of the report. The report is prepared by choosing any repeated data item to constitute a group. Click ***Next*** when the grouping level is added and defined.

(vi) Access responds by requiring the designer to specify the sort order based on any of the fields contained in the report. The records may be sorted up to four fields by specifying either ascending or descending order for each field. After specifying the sort order, click ***Next*** or specify the summary values to calculate. The summary values are sum, average, minimum and maximum. Once summary values are specified, click ***OK***, followed by click ***Next***.

(vii) Report wizard responds by requiring the designer to choose the report layout (stepped, block, outline and align left) and its orientation (portrait and landscape). Click ***Next*** after specifying the layout and orientation.

(viii) Report wizard prompts the designer to choose a particular style of report from among six styles: bold, casual, compact, corporate, formal and soft-gray. After choosing a suitable style for report, click ***Next***.

(ix) Report wizard prompts the designer to specify the title of report being designed. Further, the designer is provided with two options: preview the report or modify its design. After exercising the option, click ***Finish***.

(x) Access presents the report in preview mode or design mode depending on which option is chosen in (i) above.

(c) *Design View* : The design view method offers greatest flexibility to the designer in designing a report. In this method, the report is designed by assembling and embedding various components from report tool box. In order to design a report by using design view, following steps are required:

(i) After selecting Reports object, double click ***Create report in Design view***. Access responds by providing a blank report object with three sections: Report/Page header, Detail and Report/Page footer as shown in Figure : 15.9.

(ii) Right click the mouse at the black spot appearing at the left of horizontal ruler of above report. Report object responds by displaying a drop down window.

(iii) Click ***Properties*** and select Record Source from ***Data*** tab. The record source turns into a combo control giving a list of various tables and queries. Choose the appropriate source of information to be presented

in the report being designed. Access responds by providing a list of fields of the selected record source. If this list does not appear or it is closed by mistake, it can be recalled by clicking at the field list icon appearing before the icon for tool box.

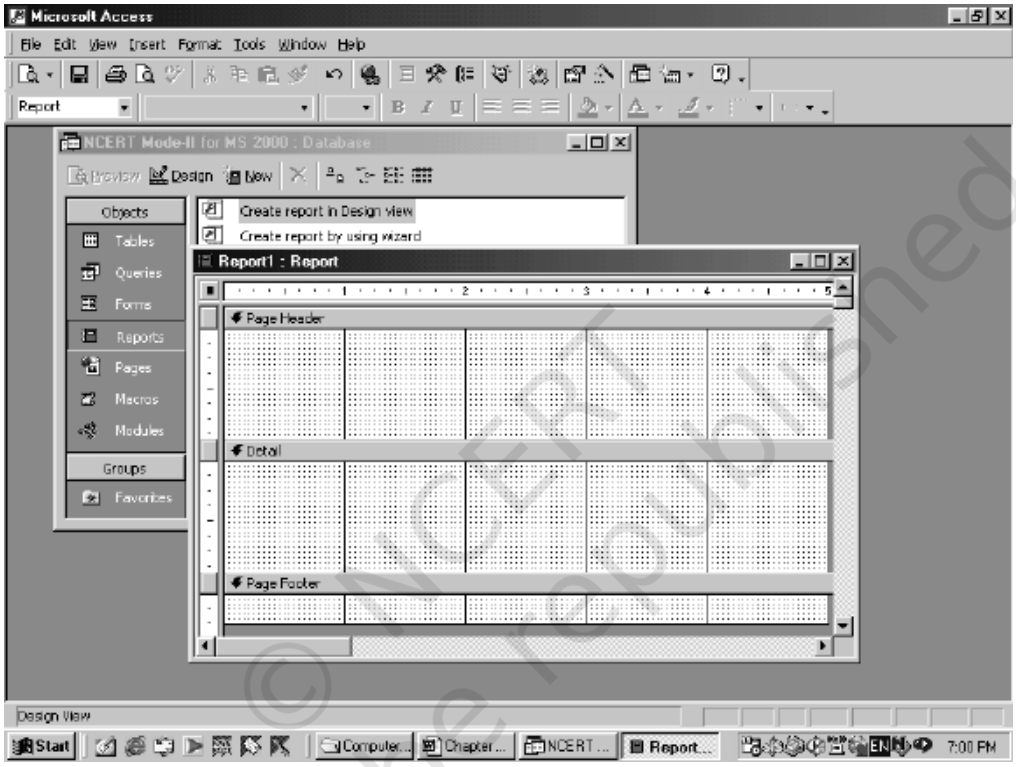


Fig.15.9 : *Window displaying design view of report*

(iv) Select the required fields from list of fields displayed as discussed in (c) above, by clicking at each of the fields to be selected while keeping the ***Ctrl key*** pressed. Drag and drop the selected fields to ***Detail*** section.

(v) The label part of each field is moved to Report/Page header and text part is accordingly aligned below their respective labels column wise. The caption of each label giving headings can be suitably modified, if required.

(vii) The vertical ruler controlling the distance between various report sections can be suitably adjusted to give a better look to the report.

The Report/Page footer bar is brought close to the fields laid out in ***Detail*** section so that the gap between records of details section is minimized.

(viii) Page headers and page footers may also be added by right click at title bar of report object, followed by click at Page header/footer.

15.5.6 Refining the Report Design

The design of the report created by any of the methods described above may be improved upon by making the following additions and modifications to the report. For this purpose, an existing report is opened in design mode.

- *Adding Dates and Page Numbers* : When an existing report is opened in design mode, the page footer of the report contains two unbound controls: the current date and ***current page number*** *of* ***total number of pages***. Both the controls may be customised according to the requirement of the designer. The date control uses = ***Now()*** function to retrieve the current date from RTC of computer. The format of date may be modified by selecting General date, Medium date, Short date or Long date from format property of this control.

  Further, when a report is created using design view method, the date and/or time and also the page numbers may be added to any of its part. The date and time is added by clicking Insert % date and time from the menu bar to open the Date and Time dialog box. After selecting and specifying the desired preferences regarding date and time, click ***OK*** to find that a text control with chosen date and time preferences is added at the top of active report section. This added text control containing date and time may be dragged and dropped in any part of the report as per requirement. Similarly, the page number is added by clicking ***Insert % page*** numbers from the menu bar to open the Page numbers dialogue box. This dialogue allows the designer to specify the format, position and alignment. The two formats are: Page N (for example Page 1) and Page N of M (for example Page 1 of 10). The position to specify is either Top of Page (header) or Bottom of Page (footer). Possible alignment, which may be specified are Centre, left, right, inside and outside.
- *Adding and Deleting Report Controls* : After a report has been designed, additional report controls may be added or deleted by the same procedure as applicable to forms. Clicking tool bar icon opens report design tool bar, which contains a set of useful controls.

  (a) After opening the report in design mode, click ***Field List*** button on report design tool bar. This results in opening the field list window.

(b) Drag the field into an appropriate section of the report. The field appears with both label and text box control. The label part gives a constant field heading while the text part provides different values of the field. These two parts are accordingly placed at the appropriate sections of the report.

(c) A field control may be deleted by selecting the control and pressing the ***Delete*** key.

- *Conditionally Formatting Report Controls* : The conditional formatting of text boxes and combo boxes in reports can be achieved in the same manner, as it applies to Forms. The conditional formatting allows the designer to apply special text formats that depend on the value of field. This facility is a useful tool to draw the attention of user or reader of report to some values of particular interest, such as amounts exceeding certain limit or unexpected balances in some accounts. In order to create a conditional formatting, following steps are required:

  (a) Open the report in design view.

  (b) Select a control and click at format on menu bar, followed by conditional formatting.

  (c) Provide the necessary conditions for formatting to occur in the same manner as already discussed while applying conditional formatting to design of Forms.

  (d) The conditional formatting is removed by re-opening the same dialog and clicking at ***delete*** button.

- *Grouping Levels and Sorting Order* : The purpose of grouping is to organise the information content of a report into categories. Sorting order is meant to arrange such information content into numerical or alphabetical order. With groupings the sorting applies to each individual group. The grouping and sorting of information, when applied together, make the report more meaningful and therefore useful to the user of the report. In order to specify the grouping and sorting order, following procedure is adopted.

  (i) Click at Sorting and Grouping icon of Report Design Tool bar (This icon is located next to icon for tool box). Immediately, Access responds by displaying the following Sorting and Grouping dialogue box.

  (ii) The LHS of this dialog box provides a list of fields or expressions that are to be used for grouping and sorting. In the above dialog box, ***Type*** field of Accounts has been chosen as the basis of grouping the information content of trial balance. The group header and footer property is set to ***Yes*** to indicate that there is separate header and footer for each group of accounts in trial balance.

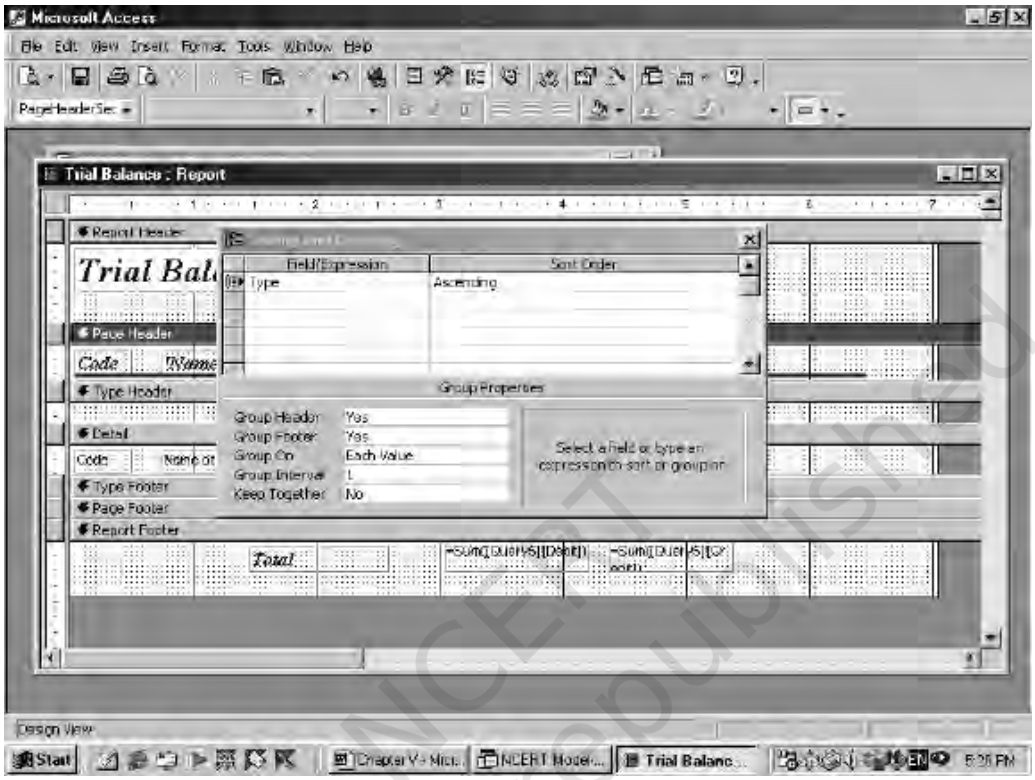


Fig. 15.10 : *Window displaying sorting and grouping dialogue box*

15.5.7 Saving and Exporting a Report

After a report is designed, it may be generated to preview its final shape. Both the design and a generated report are saved for future use and reference. The generated report may also be exported for use by others, as described below:

(a) *Saving and Exporting Report Object in Access* : The design of a report is saved in Access as report object by assigning a particular name. The report object, when opened in access by click action generates the desired report as per design specification. The design may also be exported to another database file of Access. This is achieved by clicking ***File % Export*** and then selecting and existing database into which the report design is to be exported. Access responds by providing a dialog box to give the name by which the exported report is saved in a selected database.

(b) *Saving as Snapshot* : After a report is created, it may be saved in such a manner so as to be viewed by others without the help of Access. This becomes possible by saving the report as a snapshot file. As a result, a high quality picture image of each page of report is created with Adobe Acrobat software. Other users of the report can then view the report and print any of its pages without being able to modify its contents. It must be ensured that this feature of saving a report as snapshot is also installed while installing the MS Office 2000 package. In order to create a report Snapshot, following steps are required :

- Select and generate a report in Database Window.
- Click ***File % Export*** from menu bar. An Export Report dialog box appears.
- Choose the folder from combo box next to ***Save in***; provide a file name; select snapshot from list control next to ***Save as*** type and click at ***Save*** button. While saving the report ensure that the auto start check box is enabled.
- The generated report is saved as a snapshot and can be supplied to others for printing and viewing without the help of the Access database environment.

(c) *Exporting to Excel* : A generated report may be exported to Excel, which is a spreadsheet package. This software package is a part of MS Office product and is generally installed while installing MS Access. A report is exported to Excel by following the same steps as have been listed above while saving a report as snapshot, except that before clicking save button in (c) above, one has to select Microsoft Excel 2000 from list control next to *Save as* type.

(d) *Exporting to MS Word* : A report generated using Access can also be exported to MS word, which is a text processing package. This package is also installed while installing MS Access, as a part of MS Office. In order to export a report to MS Word, the following steps are required :

(i) Select and generate a report in Database Window.

(ii) If print preview tool bar is absent in Access window, Click ***View % Tool bars % Print preview*** from menu bar of Access. Access responds by providing print preview tool bar for reports.

(iii) Click at right corner of icon for Official Links. There are three options in the list: Merge It with MS Word, Publish It with MS Word and Analyse It with MS Excel.

(iv) Click ***Publish it with MS Word***, which is also the default option.

(v) The generated report is exported to MS Word package and can be dealt with like any other document created using MS Word.

(e) *Printing a Report* : A generated report may also be printed by taking the following steps provided a printer attached to the computer is installed.

(i) Choose ***File*** from menu bar **% *Print***

(ii) Access responds by providing a print window, which allows the user to select a printer, the number of copies to be printed and also the range of pages to be printed.

(iii) Properties button is clicked to define print quality under set-up tab and orientation under paper tab. Two-sided printing may also be obtained if the printer supports this feature.

(f) *E-Mailing a Report* : A report generated by Access may also be sent using E-Mail facility, provided the computer system has Internet facility and is connected to the Mail Server of the Internet Service Provider (ISP). In order to send a report using E-mail facility, following steps are required :

(i) Select and generate a report in Database Window

(ii) Click at ***File % Send-To % Mail recipient*** from Menu bar of Access. A Send dialog box appears with various options for choosing the Format: Microsoft Excel, HTML, Snapshot format, Rich Text format, etc.

(iii) Choose an appropriate format and click ***OK.*** Access responds by providing an E-Mail composition window.

(iv) Fill up the details regarding E-mail address of recipient and others to whom copy of report is to be sent; provide a subject to E-mail and click at Send button. The report gets dispatched to the mailbox of the recipient of E-mail.

Test Your Understanding

Fill in the blanks

(a) Reports, the need for which is not anticipated is called .....reports.

(b) .....query does not involve use of any query function to produce a summary of data.

(c) ..... query prompts the user to enter criteria for selecting a set of records.

(d) .....clause is used to specify the fields to display data or information.

(e) ..... is meant to include page number, data and time of report.

(f) The purpose of ..... is to organise the information of report into categories whereas ..... arranges information into numerical or alphabetical order.

(g) When saved as ....., the contents of reports can not be modified by the user.

15.5.8 Designing Accounting Reports using Access

Financial Accounting Reports such as Cash book, Bank book, Ledger Accounts and Trial Balance may be generated in Access by adhering to report generation process. The exact process in the context of each of these reports is described below :

Trial Balance

The Trial Balance is one of the accounting reports, which provides the net amount by which each account, during a given period of time, has been debited or credited. The format of a typical trial balance is as given below :

Trial Balance

Account Title	*L.F.*	*Debit* Amout *Rs.*	*Credit* Amount *Rs.*
Total			

Fig. 15.11 : *Format of trial balance*

To produce a trial balance, it is necessary to retrieve a set of processed data records each of which provides information on Code (or Account Number), Name of Account (or Particulars), Debit balance and Credit balance with reference to a each account. In order to find net balance corresponding to every account along with its identity, following steps are taken :

(i) To find the total amount by which every account has been debited;
(ii) To find the total amount by which every account has been Credited;
(iii) To find a collective record set of accounts with their debit and credit totals;
(iv) To find the net amount with which every account has been debited or credited; and
(vi) To find the record set which consists of Account code, name of Account, Debit and Credit Amount.

Above steps to produce trial balance are transformed into a series of SQL statements, which vary according to the database design. The details of the above procedure along with the relevant SQL statements need be explained in the context of the three Models as given below :

Model-I : The following series of SQL statements retrieve a record set for producing trial balance when database design for Model-I is used.

(a) *To find the total amount by which the accounts have been debited* : In order to ascertain the total amount by which every transacted account has been debited, the SELECT clause need to have two fields: *one* code to identify the transacted account and *another* to generate the total by which such account has been debited. This is achieved by using Debit field of ***Vouchers*** table and finding the sum of amount corresponding to each of the transacted accounts. The FROM clause relies upon ***Vouchers*** table to get the data source. The GROUP BY clause specifies the field on the basis of which grouping of record set is formed. This grouping is necessary in SQL when aggregate query is used to generate summary information. The summing of amount is obtained by using aggregate function, Sum( ). This function, as already explained, uses a field with data type Number, as an input argument and returns its sum as output. Accordingly, the following SQL statement is formed :

SELECT Debit **AS** Code, **Sum(**amount**) AS** Total
FROM vouchers
GROUP BY debit;

In the above SQL statement, the GROUP BY clause retrieves the rows of vouchers table accounts-wise because the debit field refers to account code. As a result, the Sum( ) computes the sum of amount of a particular debit account and reports against Debit account of SELECT clause. This SQL statement is saved as Query 01for its subsequent use. The total of debit amount in this query is given by Total field with positive amounts.

(b) *To find the total amount by which the accounts have been credited* : In order to ascertain the total amount by which every transacted account has been credited, a query similar to that in (a) need be formed, except that the Debit field in SELECT and GROUP BY clause is substituted by Credit field. The sum of amount generated by sum(Amount) is multiplied by -1 so that the final amount assigned to Total field is always negative. This is because the amount of credit must be a negative amount if amount of debit is taken as positive. The purpose of using negative values is to differentiate between debit and credit totals for each account and also to facilitate the simple arithmetic summation for obtaining the net amount. Accordingly, the following SQL statement is formed :

SELECT Credit **AS** Code, **Sum(**Amount**)***(-1) **AS** Total
FROM vouchers
GROUP BY Credit;

This SQL statement is saved as Query 02 to be used as source by next query.

(c) *To generate a collective record set of accounts with their debit and credit totals* : Every transacted account that has been debited (or credited) only appears once in this collective record set. However, those transacted accounts that have been debited as well as credited appear twice in this record set: once with a positive amount and thereafter with a negative amount. This collective record set is generated by executing a UNION query between Query 01 and Query 02.

```
SELECT*
FROM Query 01
UNION SELECT*
FROM Query 02 ;
```

This SQL statement is saved as Query 03 for further processing of its resultant record set.

(d) *To generate the net amount with which an account has been debited or credited* : Once the records of account codes with debit and/or credit totals have been collected, the next logical step is to find out the net amount by which such accounts have been either debited or credited. This is accomplished by forming another aggregate query in which FROM clause uses Query 03 as the data source. The sum of Total for each Code of data source, provided by Query 03, results in computing net amount for every account. Accordingly, the following SQL statement is formed to generate a list of account codes with their respective balances: positive or negative.

```
SELECT Code, Sum(Total) AS Net
FROM Query 03
GROUP BY Code;
```

A positive net amount implies a debit and negative amount means a credit balance corresponding to an account code. This is because in Query 02, the total of credit amount has been made to appear as negative. This query is saved as Query 04 for its subsequent use in generating record set for trial balance.

(e) *To find that record set which consists of account code, name of account, debit amount and credit amount* : Every row of a trial balance report consists of Account Code, Name of Account, Debit Amount and Credit Amount. The Debit Amount and Credit Amount are mutually exclusive. Such rows are obtained by generating a record set based on the following SQL statement.

```
SELECT a.Code, b.name AS [Name of Account], IIF
(a.Net>0,a.Net,null) AS Debit,
IIF (a.Net<0,abs(a.Net) ,null) AS Credit
FROM Query 04 AS a, Accounts AS b
WHERE a.code = b.code ;
```

In the above SQL statement, the results of Query 04 and data stored in Accounts table has been used. The SELECT clause of this SQL statement has two computed fields as explained below :

- IIF(a.Net>0,a.Net,null) AS Debit: According to IIF( ) function, if the net amount exceeds zero, it is displayed as Debit, otherwise nothing appears in Debit field.
- **IIF**(a.Net<0,abs(a.Net) ,null) AS Credit: According to IIF( ) function, if the net amount is less than zero (implying negative), it is displayed as Credit, otherwise nothing appears in Credit field.

Besides, the other two fields: Code and Name, of SELECT clause are retrieved from Query 04 and Accounts table respectively. This SQL statement is saved as Query 05 for providing the necessary information content for Trial Balance Report.

Model-II **:** The following series of SQL statements retrieve the record set for producing trial balance when database design for Model-II is used. In addition to this, the accounts have been categorised within the trial balance according to the Account Type: Expenses, Revenues, Assets and Liabilities.

(a) *To find the total amount by which the accounts have been debited* : The transacted accounts in design of Model-II have been stored in AccCode of VouchersMain and Code of VouchersDetail. The following SQL statement is formed to generate the relevant information from VouchersDetails.

```
SELECT Code, Sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 0
GROUP BY Code ;
```

Similarly, the following SQL statement is formed to generate the required information from VouchersMain table.

```
SELECT AccCode As Code, sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 1
GROUP BY AccCode ;
```

Both the SQL statements are meant to extract similar sets of records, but from two different sources. Therefore, the resultant record set of these SQL statements have been horizontally merged using UNION clause as shown below:

```
SELECT Code, sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 0
GROUP BY Code
```

```
UNION ALL
SELECT AccCode As Code, sum(amount) AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno = VouchersDetails.Vno
WHERE Type = 1
GROUP BY AcCode ;
```

The above SQL statement is saved as Query101for its subsequent use. The total of debit amount in this query represents the Total with positive amounts.

(b) *To find the total amount by which the accounts have been credited* : In order to ascertain the total amount by which every transacted account has been credited, a query similar to that in (a) need be formed. This is achieved by substituting Debit field in SELECT and GROUP BY clause by Credit field and the sum of amount generated by sum(Amount) is multiplied by-1 so that the final amount assigned to Total field is always negative. Accordingly, the following SQL statement is formed :

```
SELECT Code, sum(amount)*-1 AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno=VouchersDetails.Vno
WHERE Type=1 GROUP BY Code, Amount
UNION
SELECT AccCode As Code, sum(amount)*-1 AS Total
FROM vouchersMain INNER JOIN vouchersDetails ON
VouchersMain.Vno=VouchersDetails.Vno
WHERE Type=0 GROUP BY AccCode, Amount;
```

I*n the above SQL statement, the sum of amount has been multiplied by* -1 to ensure that the amount of credit is always negative just as amount of debit is taken as positive. This query is saved as Query102 for its subsequent use.

(c) *To find a collective record set of accounts with their debit and credit totals*: A collective record set is generated by forming a union query between Query101 and Query102 to ensure that the debit and credit amount with respect to each account becomes available for generating the net amount. Accordingly, the following SQL statement is formed.

```
SELECT*
FROM Query101
UNION Select*
FROM Query102;
```

The above SQL statement causes horizontal merger of record sets returned by Query101 and Query102. This SQL Statement is saved as Query103 for its subsequent use in next query.

(d) *To find the net amount with which an account has been debited or credited*: To generate the net amount, an SQL statement similar to Query04 (designed for query (d) of Model-I) above, is formed as shown below, except that its source of data is Query103 instead of Query 03.

```
SELECT Code, Sum(Total) AS Net
FROM Query103
GROUP BY Code;
```

This query is saved as Query104 for its subsequent use in generating a record set, giving details of information for trial balance.

(e) *To find the record set which consists of Account code, Name of Account, Debit Amount and Credit Amount* : This query, which is meant to provide relevant information to the trial balance report, is similar to Query 05 (designed and discussed in (e) of Model-I). Accordingly, the following SQL statement is formed by changing the source of data from Query 05 to Query105 as shown below :

```
SELECT a.Code, b.name AS [Name of Account], IIF(a.Net>0,a.Net,null) AS
Debit, IIF(a.Net<0,abs(a.Net) ,null) AS Credit FROM Query104 AS a,
Accounts AS b/
WHERE a.code = b.code;
```

In above SQL statement, the results of Query104 and data stored in accounts table has been used. This SQL statement is saved as Query105 for providing source of information to Trial Balance Report.

Trial Balance with Sorting and Grouping levels : In order to prepare a trial balance with all the account duly grouped by and sorted within category of accounts, two additional queries (f) and (g) are required.

(f) *To find the record set of accounts with their category and category ID :* Accounts table is related to AccountType table vide Type field. The following SQL statement, using INNER JOIN clause, is formed to retrieve the relevant fields for various accounts.

```
SELECT Accounts.Code, Accounts.Name, Category, CatId FROM Accounts
INNER JOIN AccountType ON
Accounts.Type = Account type.CatId;
```

This SQL statement is saved as Query 106 for its subsequent use in next query.

(g) *To find the record set consisting of Account Code, Name of Account, Debit Amount and Credit Amount along with category details* : This query, when compared with (e) above, reveals that two additional fields: **Category** and

CatId are required. Accordingly, the SQL statement stored as Query105 is modified by substituting Accounts table with Query106 to form the following Statement.

```
SELECT a.Code, b.name AS [Name of Account],
IIF(a.Net>0,a.Net,null) AS Debit, IIF(a.Net<0,abs(a.Net) ,null) AS Credit,
Category, CatId
FROM Query104 AS a, Query106 AS b
WHERE a.code = b.code ;
```

This SQL statement is saved as Query107 to provide information details for designing trial balance with grouping and sorting of the accounts.

15.5.9 Procedure in Access for Designing a Simple Trial Balance

The Trial Balance is generated using the ***Design View*** method by following the steps listed below :

(i) Select ***Reports*** from objects list provided by LHS of Database Window and click at ***New*** object button of tool bar. Access responds by displaying the New Report Window as shown in figure 15.8 Choose Design View from list of methods and Query 05 from combo control meant to provide data source to the report. Click ***OK*** after choosing method and data source of report.

(ii) Access responds by displaying a blank report design divided horizontally into three sections: Page Header, Detail and Page Footers. Besides, a list of available fields of Query 05 is also provided for embedding on to this blank design of report.

(iii) Alternatively, double click at ***Create report in design view***. Access respond by displaying a blank report design duly divided into three sections as stated above. Right Click at the left most corner point of report design where horizontal and vertical rulers converge. Click at Properties of report and select Data tab to define the record source as Query 05. Immediately, there appears as list of available fields of Query 05 so as to be placed on to blank design of report.

(iv) Right click at any part of the report design and choose Report Page Header and Footer. Access responds by providing two more sections: Page Header and Page Footer.

(v) Click at the icon for tool bar and pick up a label control to be placed at Page Header Section and assign set its caption property to ***Trial Balance***, Font Size to 16, Font colour to Blue, Text align to Left and Font weight to Bold.

(vi) Select all the fields of Query 05 by clicking at every field while keeping the ***Ctrl key*** pressed. Drag and drop the selected fields on Details section. It may be noted that each of the dropped fields has two controls: Label and Text. The former gives caption and the latter provides the data content.

(vii) Select the label controls of all the four fields by clicking at each while keeping the ***Shift Key*** pressed. Right click at selected label controls and choose ***cut***. Place the mouse at Page Header section and ***paste*** these controls.

(viii) Re-arrange these label controls to appear as headings of columns for trial balance as: Code, Name of Account, Debit and Credit. Select all these label controls and right click to choose properties. Access provides Properties of these controls. Choose format tab and set the Font weight Property to Bold; Font Size to 10; Font colour to Blue and Text align to Centre.

(ix) Align the Text controls in Detail section to appear just below each of the respective label controls appearing in Page Header section.

(x) Select the Text controls and Debit and Credit field and modify their properties by setting Decimal Places to Zero and Format to Standard.

(xi) Pick up a label control from tool box by click action and place at Report Footer section, at the area vertically below the column "Name of Accounts" and give the caption "Total". Set its Text align property to Centre, Font weight property to Bold and Font Size to 10.

(xii) Pick up a text control and place it at Report Footer section at the area vertically below Debit column. Set its Record source property as expression given below :

= Sum ([Query 05]![Debit])

The expression is written by clicking at (...) to call the expression pane. The expression [Query 05]![Debit] within Sum( ) function refers to Debit field of Query 05.

(xiii) Pick up another text control and place it at Report Footer section at the area vertically below Credit column. Set its Record source property as expression given below.

=Sum ([Query 05]![Credit])

The expression is written in the manner as it applies to sum of debit column. The expression [Query 05]![Credit] within Sum( ) function refers to Credit field of Query 05.

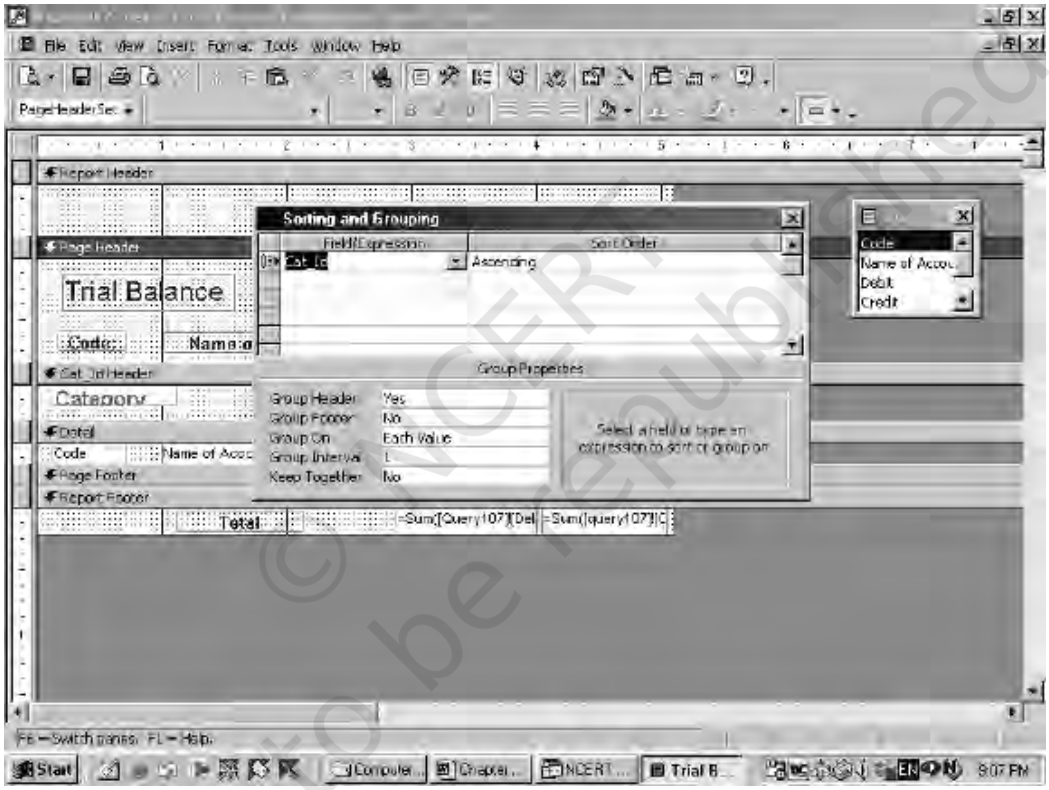
The report design prepared above is saved as Trial Balance by Design. The Trial Balance report design appears on the RHS of Database Window as object under Reports.

15.5.10 Designing of Trial Balance with Sorting and Grouping

To design a trial balance with grouping and sorting of accounts, the following additional steps are required.

(i) Copy the trial balance design as created above and paste it with different name say "Trial balance with Grouping". Open this copied report design for modification in design view to incorporate the grouping and sorting of accounts in trial balance report.

Fig 15.12 : *Window displaying sorting and grouping dialog*



(ii) Change the data source property of report design by right click at the top left corner of report design % click at properties % Choose Tab and set the Record source property as Query107.

(iii) Modify the Record source of Text controls for sum of debit and credit columns to replace existing expressions by

= Sum ([Query107]![Debit]) ..... for Debit

= Sum ([Query107]![Credit]) ..... for Credit

(iv) Right click at report design % click at sorting and grouping. Access responds by providing a window for sorting and grouping as shown in figure 15.12

(v) Define the basis of grouping as CatId in field/expression and its sort order set to ascending. Set the Group Header property to ***Yes.*** Access responds by inserting CatId Header section in report design.

(vi) Click at field list icon and drag and drop category field in CatId Header section. Set its Font Size property to 10, Fore Colour property to Dark Green and Font Weight property as Bold.

Save the modifications in the above report design. The trial balance report is generated by double click at this or the previous object. The generated trial balance may be saved or exported as desired.

Key Terms Introduced in the Chapter

- MS Access
- Accounting Report
- Compound Vouchers
- Database Management System
- Transaction Vouchers
- Queries

Summary with Reference to Learning Objectives

1. *Accounting Reports* : A report displays information that is acquired from data processing and transformation in an organised manner. Reports tend to reduce the level of uncertainty associated with decision-makers and also influence their positive actions. The output of the computerised accounting system are accounting reports. Financial accounting reports such as Cash book, Bank book, Ledger, and Trial Balance may be generated in Access by adhering to report generation process.
2. *Using Access for Producing Reports* : In Access, the reports are created by designing a report, identifying its information requirement, creating the queries in SQL to generate such information so that the final SQL statement provides the record set of information to the report design. Different Models of database design require different sets of SQL statements to produce different types of reports.
3. *Queries Access* : There are several types of queries in Access that may be used to generate information. Such queries are called select queries because they are used to select records from the given set of records. There are three ways in which these queries may be created in Access: Wizard, Design View and SQL View method.
4. *Designing Reports in Access* : A report in Access may be designed in three ways: Auto Report, Wizard and Design View method. A SQL statement (or query) is capable of displaying records containing fields from across a number of data tables. A typical report in Access has the structure that consists of Report header, Page header, Group header, Details, Group footer, Page footer and Report footer.

Questions for Practice

Short Answers

1. State what do you understand by accounting reports.
2. What do you mean by programmed or casual reports?
3. With the help of an example, briefly state the meaning of parameter queries.
4. Briefly state the purpose of functions in SQL environment.
5. Briefly explain in steps the method of creating a query, using wizard.
6. List the structure of a good report created in Access.
7. List the ways to refine the design of a report.
8. Briefly explain the purpose of grouping and sorting of the data as a means to refine a report.
9. What do you understand by saving a report as snapshot?
10. State the procedure for creating ledger in MS Access.

Long Answers

1. Describe and discuss the procedure of creating the receipts side of a cash book.
2. Discuss the concept of accounting reports? Explain the three steps involved in creating such reports.
3. Discuss with a set of inter-related data tables, the basics of creating queries in MS Access?
4. Briefly explain the set of SQL statements to produce the receipts side of a cash book for Model-I.
5. Describe in steps the design view method to create a query in MS Access?
6. Discuss the SQL view method of creating a query?
7. Describe the ways to refine the design of a report.
8. Explain the data base design for Model-I for producing the receipts the series of SQL statements for producing the payment side of cash book for Model-II.
9. Describe the series of SQL statements to produce trial balance data base design for Model-II is used.
10. Using Model-III discuss the series of SQL statements to produce a trial balance up to a particular date.

Project Work

1. Payroll Accounting: Using the database design given in Exercise of Chapter-IV, as Project No: 1, you are required to generate the portion of payroll according to the specified format under MS Access environment.
2. Financial Accounting: Write the SQL statements for each of the following queries separately by using database design of accounting specified as Model-I, II and III in Chapter-IV.
   (a) List the transactions details of Accounts, which have been debited during the period April 01, 2014 to September 30, 2014.
   (b) List the transactions details of accounts which have been credited during the month of August 2014.
   (c) Find the total expenses incurred during the period September, 2014.
   (d) List all the transacted accounts with the amounts by which they have been debited and also the amount with which they have been credited.
   (e) List the amount of expenses authorised by each of the employees.

3. Inventory Accounting: Using the database design developed in Exercise of Chapter-IV, for Project No: 2, you are required to generate Statement of closing stock in the following format by assuming that all goods are sold at a profit of 25% on purchase price.

Statement of Closing Stock

Particulars		*Purchases*		*Sales*		*Balance*	
Code	*Item Name*	*Qty*	*Amount*	*Qty*	*Amount*	*Qty*	*Balance*

4. Inventory Accounting: Using the database design developed in Exercise of Chapter-IV, for Project No: 2, Write the SQL statements for each of the following queries :
   (a) List out the Invoice No, Date and amount of sales made during the month of October, 2014.
   (b) Make a list of Invoice No, Date and amount of Purchases during the period April 01, 2014 to October 31, 2014.
   (c) List items wise the quantity sold during the month of September 2014
   (d) Find the Minimum and Maximum rate at which each item of goods has been purchased during the period April 01, 2014.
   (e) Make a list of physical quantity of each item in stock.

Checklist to Test Your Understanding

(a) Casual
(b) Simple
(c) Parameter
(d) SELECT
(e) Design view
(f) Sorting
(g) Snap shot

APPENDIX

Description of Commonly Used Functions in Access

There are three types of functions that are used to set the Control Source property of calculated controls and/or to form part of calculated field expression in SQL statement. A brief description of the commonly used functions is below :

A-1. Domain Aggregate Functions

These functions are used to perform calculations based on values in a field of a table or query. Criteria to select the set of records in the table or query that is desired to be used for calculations may also be specified. The criteria, if not specified, imply that all the records of the table or query specific to the field are used for computation. All the domain aggregate functions use the same syntax as is given hereunder :

DFunction ("FldName", "TblName" or QryName", "SrchCond")

Wherein DFunction refers to a named domain aggregate function. A brief description of its input arguments is given below:

FldName : It refers to the name of field that is to be searched in a table or query, which is specified as an argument.

TblName (or QueryName) : It refers to the name of a table or query that contains the field specified as second input argument.

SrchCond : It refers to the search condition on the basis of which the relevant record is searched.

Some of the important domain aggregate functions have been described as below :

(a) DLookup : This function is meant to look up information that is stored in a table or query, which is not the underlying source of Access Form or Report. It is used to set the Control Source property of a calculated control to display data from other table or query. Consider the following example:

DLookup ("Name", "Accounts", "Code = '110001'")

In the above example, this function has been applied to search the name of account (in Accounts table) whose code is '110001'.

(b) DMax and DMin : These functions are used to retrieve respectively the maximum and minimum values in the specified field. Consider the following example :

DMin ("Amount", "Vouchers", "Debit = '711001'")

Dmax ("Amount", "Vouchers", "Debit = '711001'")

In the above examples, the amount of minimum purchase transaction and maximum purchase transaction is retrieved and reported. It may also be noted that '711001' is the code of Purchase account in Accounts table

(c) DSum : This function computes and returns the sum of the values in the specified field or expression. For Example, in a table : ***Sales*** that contains

ItemCode, Price and Quantity as fields, the total amount of *sales* may be computed by using the DSum () function as follows :
DSum ("Price*Quantity", "Sales")
However, if the total sales is to computed for a particular item coded as 1678, the DSum () function shall be applied as follows :
DSum ("Price*Quantity", "Sales", "ItemCode = 1678")

(d) DFirst and DLast : These functions are used to retrieve respectively the values in the specified field from first and last physical records.
Consider the following application examples :
DFirst ("Name", "Accounts")
DLast ("Name", "Accounts")
In the above examples, the name first and last account that physically exists in Accounts table is retrieved and reported.

(e) DCount : This function is meant to compute the number of records with non-null values in the specified field. Consider the following application example :
DCount ("*", "Accounts")
In the above example, The number of records in accounts table are counted and reported by DCount () function.

A-2. SQL Aggregate Functions

The SQL aggregate functions have the functionality similar to that of domain aggregate function. However, unlike domain aggregate functions, these functions cannot be called directly into controls used in Forms and Reports of Access. These functions are used in SQL statements that provide the underlying record source of Forms and Reports. All these functions, when used require the GROUP BY clause in SQL statement :

(a) Sum : This function is used to compute and return the sum of a set of values. For Example, consider the following SQL statement that has been used in Chapter-V to prepare the underlying information source of Trial Balance (Model-I.).

```
SELECT Debit As Code, Sum (Amount) As Total
FROM VOUCHERS
GROUP By Debit ;
```

In the above SQL statement, Sum () has been used to compute the total amount by which the transacted accounts have beeen debited.

(b) Min and Max : These functions are used to retrieve respectively the minimum and maximum of value set with respect to field or query expression. For Example, the following SQL statement is capable of returning the amount of minimum and maximum sales transaction in Model-I :

```
SELECT Min (Amount) As MinSales, Max (Amount) As MaxSales
FROM Vouchers
WHERE Credit = '811001' ;
```

It may be noted that the sales account that is coded as '811001' is credited as and when a sales transaction is recorded.

(c) Count : This function counts the number of records returned by a query. The number of times a sales transaction has occurred and recorded in books of accounts can be known by executing the following SQL statement.

SQL statement.

```
SELECT count (*)
FROM Vouchers
WHERE Credit = '811001'
```

In the above SQL statement, the Credit field stores the account code of sales when a sales transaction occurs. The WHERE clause restricts the number of records returned by the above SQL to those in which credit field has the account code of sales. Accordingly, the count () function returns the count value of records returned by the above SQL statement.

(d) First and Last : These functions are meant to retrieve the first and last record of a value set pertaining to a field or query expression.

A-3. Other Functions

(a) IIF : The purpose of this function is to provide a value to the field from a mutually exclusive set of values. Its syntax is as given below :
IIIF (<Condition>, Value-1, Value-2)
Wherein <Condition> refers to any logical expression in which a comparison is made by using following comparison operators :
= equal to
<less than
>greater than
<= less than or equal to
>= greater than or equal to
The condition formed by the above comparison operators is evaluated to result into TRUE or FALSE.
<Value-1> This value is returned by IIF() function to the field, if the condition turns out to be TRUE
<Value-2> This value is returned by IIF() function to the field, if the condition turns out to be FALSE
Example : Suppose a field Type is to return the string of characters "Debit" when its value is 0 and "Credit" when its value is 1, IIF() function is used as shown below :
IIF (Type = 0, "Debit", "Credit")

(b) Abs : The purpose of this function is to return absolute value, This function receives a numeric value as its input argument and returns an absolute value.
Consider the following examples on use of Abs ( ) function :
When – 84 is given as input argument to Abs(– 84), it returns 84
When 84 is given as input argument to Abs(84), it returns 84

(c) Val : The purpose of this function is to return the numbers contained in a string as a numeric value of appropriate type. Its Syntax is **Val**(string)
The string argument of the above Val( ) function is any valid string expression. The Val( ) function stops reading the string at the first character that cannot be recognised as number. For example, Val("12431") returns the value 12431 by converting the enclosed string of numerals into value. However, Val ("12,431") returns the numeric value 12 because comma after 12 in the enclosed string of characters in Val ( ) function is not recognised as number.