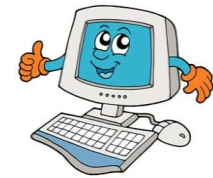


# 10

# Working with Tables



As discussed in the previous chapter, we all have to take many decisions daily and for that we need one or the other type of information. To get the correct information at right time, we have to prepare database and perform various operations like inserting data, editing the previously entered data, delete unnecessary and wrong data or arrange the data as per requirement. In previous chapter, we have already discussed how to create database and various tables in Base. Now, let us learn how to do various operations on the tables created by us.

## Inserting data in the table

To insert records into the table, we must open the table first. To open the table, double click on the icon that has the required table name in Tables Pane of Database Window. Alternatively right click on the desired table and select the *Open* option from the sub menu visible.

Let us open the Supplier table we had created using the Design View. When we open the Supplier table, its structure will be displayed in the Datasheet View as shown in figure 10.1.

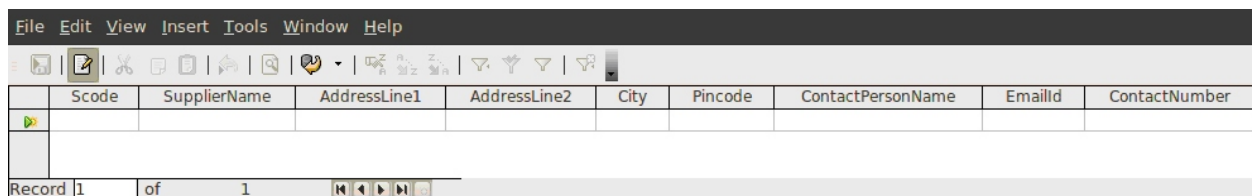





Figure 10.1 : Initial data entry screen

Here, we can see that the screen displays the field names in horizontal line. The line consisting of field names is known as Title line. We can see that it can show only some limited fields that can fit on the screen. To see the remaining fields, we have to use the scroll bar given at the bottom of the screen. Below the Title Line, there is a row consisting of empty boxes. The job of filling all these empty boxes with appropriate data is known as Data Entry. In front of the first field there is a small box that contains a pointing arrow sign. This box is known as 'Record Selector Box' and the sign in it is referred as 'Record Selector Icon'. Some of the Record Selector Icons and their functions are as mentioned below:

The black pointing arrow icon  is known as the **record pointer** that indicates a current record (current row) of the table at any given moment of time. This means that if we start typing, changes in the field value will appear in a current record.

The green icon with flash  is the end of the table mark. It is displayed in the next to the last record in the table. To add new record in Datasheet View, scroll to the last row, then click. By

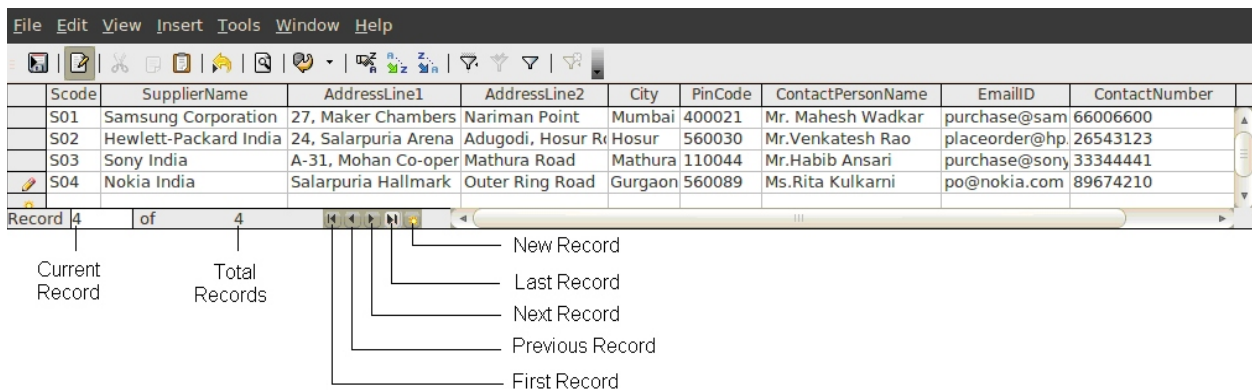
doing so, our cursor will be positioned in the field and the icon will be changed to the black pointing arrow (▶).

The pencil icon  is an Edit Icon. When we click on any of the field to edit the data, the black pointing arrow icon will be replaced with this pencil like icon. This icon will be displayed, when we have made changes to a record that have not yet been saved. If we move the cursor to another record, then correction will be saved, and if we press ESC key then the correction will be cancelled and the original contents will be restored.

Observe the bar at the lower left portion of figure 10.2, there is a word ‘Record, then a numeric value ‘4’ in the box and at the end word ‘of 4’. This bar is known as **Navigation Bar**. It contains some navigation buttons to scroll the records vertically. It also lets us know the current position of the record pointer. The meaning of ‘Record 4 of 4’ is that we have total 4 records in our table and at present our cursor is on the fourth record.

Let us fill various fields with the relevant data discussed in previous chapter. The data entry in Base is controlled and restricted based on the data type and field property that we have defined for each field. Observe that after entering data in the last field of the current row, the cursor automatically comes to the first field *Scode* of the next row.

In case of the Boolean type field a square box will be shown in the field. If we click on it, it will be marked with a tick sign. If the box is marked with tick mark sign it means that the value of the field is true, otherwise false. After entering some records our screen may look somewhat as shown in figure 10.2.



**Figure 10.2 : Datasheet view after inserting records**

### Editing Records in the Table

The data once entered may have to be edited for one of the following two reasons:

1. The data entered during data entry is incorrect.
2. After correct data entry, there is a change in the value of the data; in this case we have to edit our record. For example, assume that we have entered the correct address of

a particular Supplier. After sometime the supplier shifts his company location. In this case though the address of that Supplier was entered correctly, we will have to edit it due the change in location.

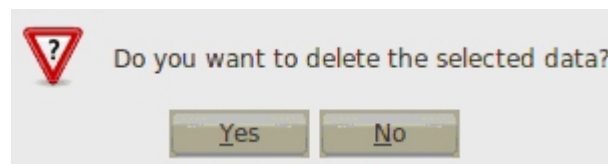
The process of correcting the data entered previously is known as **Editing**. To do edit operation, we have to open the table and simply place the cursor on the field value that we want to edit and make the desired correction.

### Deleting Records from the Table

To keep our database precise and meaningful, it is very much necessary to delete unnecessary or incorrect records from the table. Doing this will provide a clear picture and at the same time it will free some disk space. To delete any record from the table, open the table and select the unwanted record. After selecting the record, we can delete it using following two ways:

1. By pressing DEL key from the keyboard or selecting *Delete* option from Edit menu.
2. By right clicking the selected record and choosing *Delete Record* option from the sub menu.

When you delete the record, Base will display a dialog box with a warning message as shown in figure 10.3.






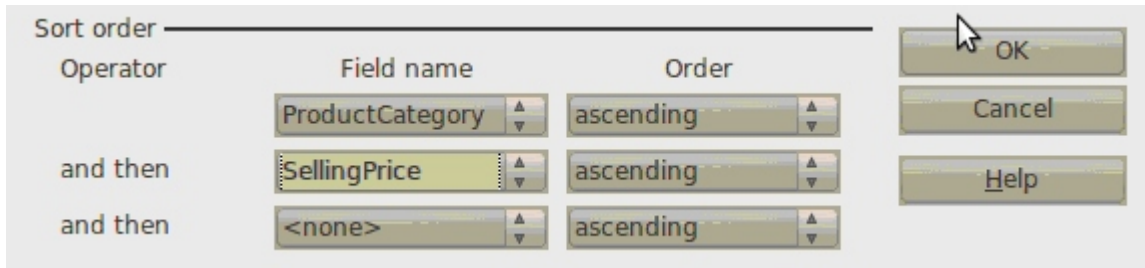
**Figure 10.3 : Record Deletion Alert Box**

From this dialog box, if we click on the Yes button, then the selected record will be deleted from the table permanently. But if we click on No button then the record will not be deleted from the table and it will be redisplayed in the view.

### Sorting Data in the Table

We create and maintain the database, because our final aim is to find the information as and when needed. Gradually the records in table will increase hence to get the information easily and speedily, it is desirable that the data in the table is arranged in some particular order. A Product table may contain thousands of records related to different products. Assume that we want to find the details of a particular product along with its price. Now in this case, if the product table is arranged in order of field Pcode i.e product code, it becomes very easy for us to find it. But, suppose we do not know the product code, then...!! Definitely, we can't find the record of that product easily and speedily. But if we have arranged the Product table in order of PCode and SellingPrice fields then it will be easy to get the desired record. So, let us see how to sort the table. Table can be sorted in one of the two ways as mentioned below:

1. Select on the field that we want to sort and then click on either Sort Ascending button  or Sort Descending button  from the toolbar, as per your needs.
2. Alternatively click on Sort button  and a dialog-box as shown in the figure 10.4 will be displayed.



**Figure 10.4 : Choice of Sort Order**

Select appropriate field value under the *Field name* dropdown box and choose required order of sorting under the *Order* dropdown box. In our case, the Product table will be sorted based on ProductCategory in ascending order first and then the data will again be sorted in ascending order of the SellingPrice field. For example all Laptops would precede all Mobiles. Within Laptops and Mobiles, the records would be arranged according to their selling price with product with lesser price listed first.

### Controlling Data Redundancy

By now, you might have become quite familiar with Base. So let us continue our discussion of designing database for our sample application. We have already designed four tables, Supplier, Product, Employee and Customer. Now let us have a relook at the table design from the point of view of data redundancy. *Data redundancy is a term used for unnecessary duplication or repetition of data.*

We have seen that the Customer table has a field called City. Modern Electronic Store would have many customers that belong to the same city. Thus if 100 customers belong to city called ‘Ahmedabad’, then for each of these customers the string ‘Ahmedabad’ would be repeated. Data redundancy thus leads to wastage of storage space and is inefficient for several reasons. For example, a change in redundant data requires changes at multiple places in a database.

To eliminate redundant data from the database, we must take special care to organize the data stored in the tables. Database designers attempt to eliminate it as far as possible by using a technique called data normalization. *Normalization is a process that suggests decomposition of single table into multiple tables, thus creating a parent-child relationship.* Thus in case of Customer table in our sample database, a new table can be created to store information about

city. Let us name this table as City. We are aware that each area within a city has a distinct pincode associated with it. A pincode can be used to uniquely identify city and area within city. Thus pincode can be used as a primary key to identify each record in the City table. Table 10.1 shows the structure of City table and its sample records.

<b>City</b>		
<b>Pincode</b>	<b>Area</b>	<b>CityName</b>
380058	Bopal	Ahmedabad
380006	Law Garden	Ahmedabad
363421	Raska	Limbdì

**Table 10.1 : City table**

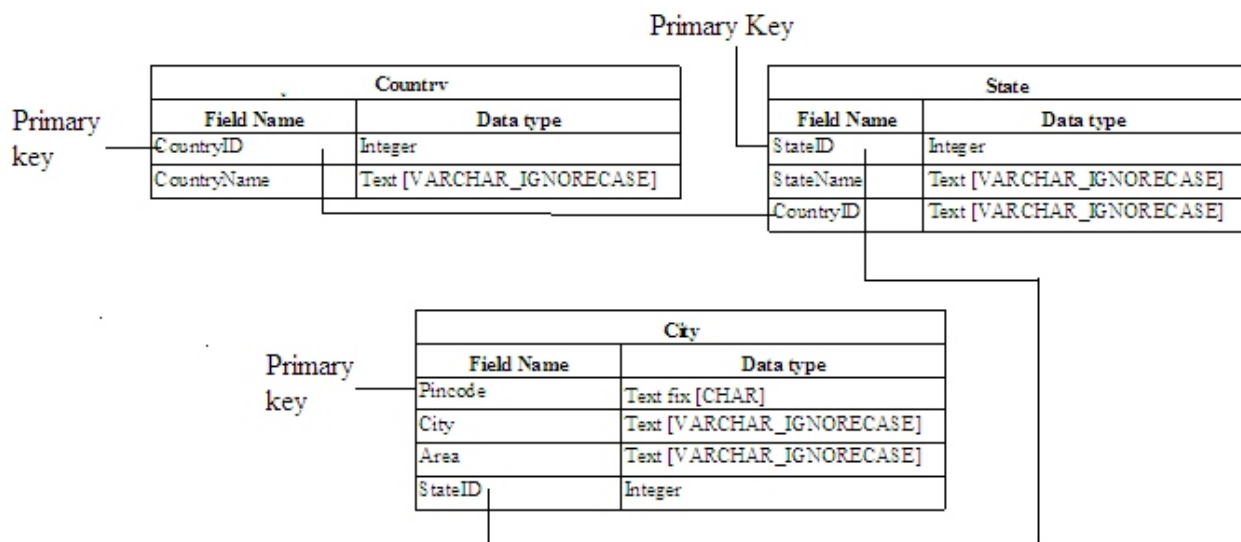
We now need to make changes in the Customer table. We can remove the City field from the Customer table. When the data of a new customer is inserted in the Customer table, we will use the pincode field to get the information regarding customer's city and area.

Similarly, the City field should be removed from Supplier and Employee tables. To maintain complete postal address information, we can also add the fields State and Country to which the city belongs. One way to maintain this information is as shown in Table 10.2.

<b>Pincode</b>	<b>Area</b>	<b>CityName</b>	<b>StateName</b>	<b>CountryName</b>
380058	Bopal	Ahmedabad	Gujarat	India
380006	Law Garden	Ahmedabad	Gujarat	India
363421	Raska	Limbdì	Gujarat	India

**Table 10.2 : City table with State and Country**

Now if we are given a pincode, the information pertaining to the location of customer can be searched from City table. But you can observe that above designed city table contains same flaw called data redundancy. Same city and state gets repeated multiple times. To avoid data redundancy, again City table has to be decomposed into multiple tables. In place of the field StateName, a field StateID should be stored and in place of the field CountryName, a field CountryID should be stored. A new table that stores information about the State and Country should be created. Figure 10.5 shows structure and relation of these tables.



**Figure 10.5 : Country, State and City Tables**

Here we have assigned CountryID, StateID and Pincodes fields as Primary keys of Country, State and City tables respectively. As seen in figure 10.5 now city, state and country tables are related with each other. Also as the pincodes would be one of the fields in Customer, Employee and Supplier tables, these tables are related with the City table. **The table which contains primary key is known as master table.** As discussed earlier, two tables are related to each other using a common field. In above example, the State and Country table is related using CountryID field. The field CountryID acts as primary key in the Country table. The field CountryID in the State table thus borrows the values from the field CountryID in the Country table. The CountryID field in the State table is known as Foreign key.

**Foreign key** can be defined as field or combination of fields whose values are borrowed from another table. When it is combination of more than one field, it is known as composite foreign key. When a master table has composite primary key, tables related with it will hold composite foreign keys.

**Note :** Foreign key field cannot contain a value that does not exist in its related primary key field.

Now if you look at Product table, product category “Mobile” or “Camera” would be stored multiple times. Thus a separate table named ProductCategory should be created. Each product category now could be assigned a unique number. This unique number can then be used to identify the category of a product. For example, Mobile could be assigned category code 1. Thus when a record for new mobile is inserted in the Product table, a value ‘1’ is to be stored in the ProductCategory field. Note that the data type of the ProductCategory field in the Product table and CategoryCode field in the ProductCategory table should be same.

Similarly, the field SupplierName in the Product table can be replaced by the field Scode that refers to supplier code. Table 10.3 lists the primary key and foreign keys of each table after modification suggested so far.

Table name	Primary Key	Foreign Key
Country	CountryID	-
State	StateID	CountryID (referencing Country table)
City	CityID	StateID (referencing State table)
Supplier	Scode	Pincode (referencing City table)
Customer	Ccode	Pincode (referencing City table)
Employer	Ecode	Pincode (referencing City table)
ProductCategory	CategoryCode	-
Product	Pcode	CategoryCode (referencing ProductCategory table) Scode (referencing Supplier table)

**Table 10.3 : Primary and Foreign keys of each table discussed in Sample database**

Let us now design other tables to store data regarding transactions like purchasing of a product by the customer. We can call it as purchase order placed by customer. A customer can either place order online or can walk into the store and purchase a product in Modern Electronic Store. An order placed by customer indicates that information related to customer, employee and product involved in the transaction has to be maintained. We cannot miss out date of order placement. Again quantity of each product is equally important. A customer may purchase multiple piece of same model. Table 10.4 shows fields of the Order table.

<b>Order</b>	
Field Name	Data type
<b>OrderID</b>	Integer
OrderDate	Date
Ccode	Text fix [CHAR]
Ecode	Text fix [CHAR]
Pcode	Text fix [CHAR]
Quantity	Integer
SalePrice	Decimal

**Table 10.4 : Order Table**

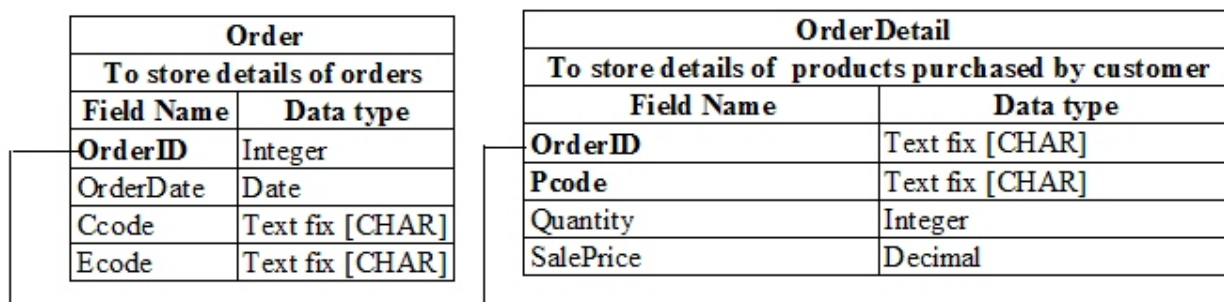
Here OrderID is designated as a primary key. Additional field is necessary in this case as no other field or their combination can uniquely determine records. All fields of Order table may have duplicates. Try to insert a record <1, 23-June-2012, C01, E01, P00000001, 2>. Now assume that customer has purchased two different products, P00000001 and P00000002. Then a new record that needs to be inserted would be <1,23-June-2012,C01,E01, P00000002, 1> .

You can observe that details like order date, employee code and customer code are repeated. If a customer purchases 10 products at a time, 10 different records are to be inserted with too much of data duplications. OrderID cannot act as primary key in that case. A combination of OrderID and Pcode is to be designated as Primary key. Solution to the discussed problem is to decompose the Order table into two tables, Order and OrderDetail. Thus fields OrderDate, Ccode and Ecode would be stored only once with OrderID uniquely identifying them in the Order table. The OrderDetail table would then contain fields OrderID, Pcode, Quantity and SalePrice. Values of OrderID will be borrowed from Order table and thus it will be foreign key field. The OrderDetail table thus contains the details which are being repeated. Table 10.5 shows sample records of OrderDetail table.

<b>OrderDetail</b>			
OrderID	Pcode	Quantity	SalePrice
O00001	P00000001	2	35000
O00001	P00000002	1	20000
O00002	P00000009	1	43000

**Table 10.5 : Sample records for OrderDetail**

Now, what should be the primary key of the OrderDetail table? As discussed earlier, field designated as a primary key cannot have duplicate values. In OrderDetail table as shown in sample records, all the fields will have duplicate values. In such situations we need to find combination of fields that will have unique values. In the OrderDetail table, combination of OrderID and PCode definitely would be unique. The product code would not be repeated in a same order. Combination of two fields can be designated as a primary key. Combination of more than one field when designated, as a primary key, is also known as *Composite Primary Key*. Figure 10.6 shows structure of both the Order and OrderDetail tables.



**Figure 10.6 : Structure of tables Order and OrderDetail**



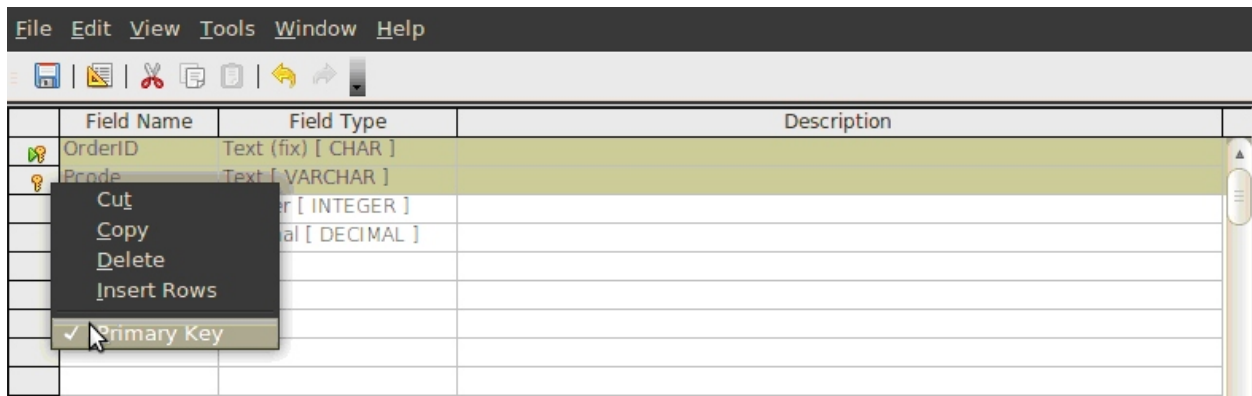
Before proceeding further let us see how can we create composite primary key.

### Composite Primary Key

To create composite primary key in OrderDetail table,

- Select row containing OrderID field.
- Press CTRL key and Select Pcode row. Both the rows will be simultaneously selected. See figure 10.7.
- Right click in selected area. A popup menu with option *Primary Key* will be displayed.

Click on *Primary Key* option and key symbol will be displayed in the left of the two selected rows.



**Figure 10.7 : Defining Composite Primary Key**

Some more tables like OrderPayment and OrderShipment are also to be designed. Figure 10.8 shows the structure of both these tables.

OrderShipment	
To record shipping address of a customer who has placed order online	
Field Name	Data type
OrderID	Integer
ShippingAddress1	Text [VARCHAR_IGNORECASE]
ShippingAddress2	Text [VARCHAR_IGNORECASE]
Pincode	Text fix [CHAR]
ShipingDate	Date
ShippingStatus	Boolean

OrderPayment	
To store details of payment made by customer	
Field Name	Data type
OrderID	Integer
PaymentID	Integer
PaymentDate	Date
PaymentAmount	Decimal
PaymentMethod	Text [VARCHAR_IGNORECASE]
ChequeDDNumber	Text [VARCHAR_IGNORECASE]
BankName	Text [VARCHAR_IGNORECASE]
BankBranch	Text [VARCHAR_IGNORECASE]

**Figure 10.8 : OrderPayment and OrderShipment tables**

The OrderShipment table would contain address where the products are to be delivered. The ShippingStatus field is used to maintain the status of the product delivery. Initially its value would be set to 'N'. Later as the product is received by customer; it has to be set to 'Y'.

The OrderPayment table would contain details of payment received from customer. A customer may be given option of payment in installments and thus in the OrderPayment table OrderID and PaymentID are designated as composite primary key. For example, for OrderID “1” PaymentID could be 1, 2, 3, 4... and so on. For OrderID “2”, PaymentID would again start from 1. Payment method could be cash, cheque, credit or debit card. The PaymentMethod field would contain one of these four options. If the payment is made by cheque, then the cheque number and name of issuing bank would also be recorded in the OrderPayment table.


Make sure that you now have the Supplier, Employee, Customer, Product, Country, State, City, ProductCategory, Order, OrderDetail, OrderShipment, OrderPayment tables with required changes in your database. In case all these tables are not present create them using either the Wizard or Design View.

### Creating Relationships between Tables in Base

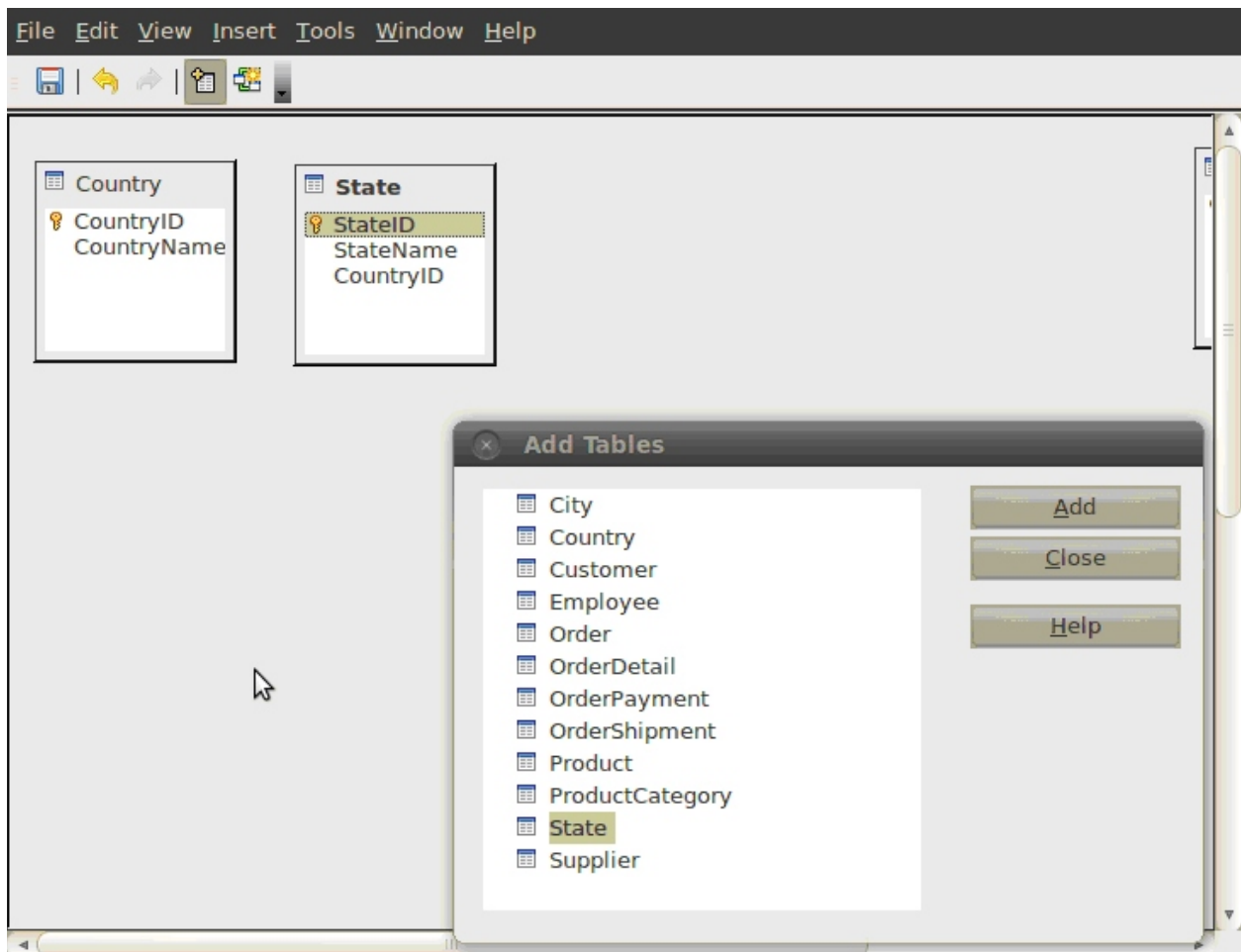
By now, you might have designed the tables we have discussed. Assume that a record of a new customer from Goregaon, Mumbai, Maharashtra is to be entered in the Customer table. Till now Maharashtra is not entered in the State table, neither the Goregaon area of Mumbai city has been entered in the City table. Insert a record of a customer with pincode value 452001. Here the value 452001 is the pincode of Goregaon area, in Mumbai city, in the State of Maharashtra. Base will accept the record entered by you. But think logically, should this record entry be allowed? No we should not allow such entry to be done in the Customer table. The reason is very simple; there is no corresponding record in the City table that that be related with this entry. To enforce such restriction, we should use the *Relationship* feature of Base. Establishing relationship between tables will restrict the user from entering garbage data in the referenced fields.

Let us learn how to establish the appropriate relationship among the different tables of the database that we have created using Base.

- Click *Relationship...* option from the Tools menu.

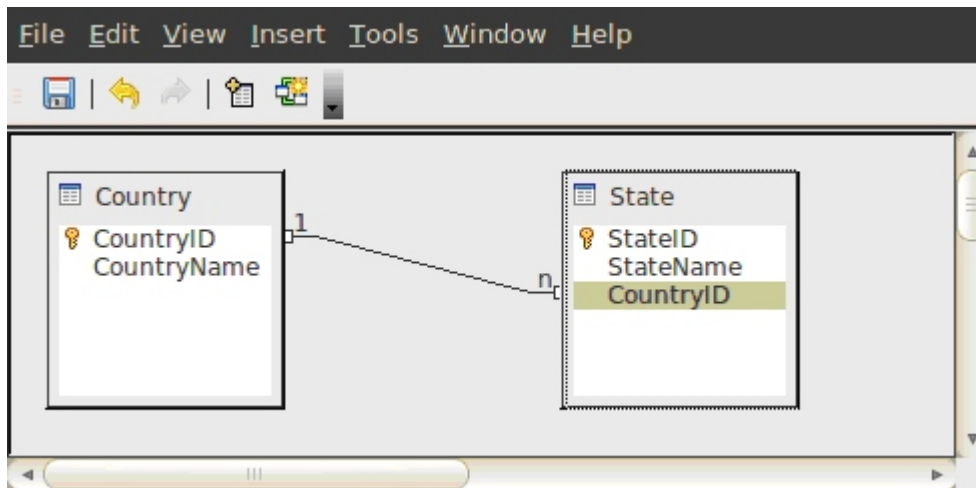
Then select *Insert* → *Add Tables*. Alternatively you can click on the Add Tables button (  ) seen in figure 10.9. Add Tables dialog box with list of tables created in the active database will be displayed.

- Select the Country table icon and click on Add button. You will find that selected table along with all its field is displayed in the background in Relationship window.
- Similarly, select the State table. The Relationship window will look as shown in figure 10.9.



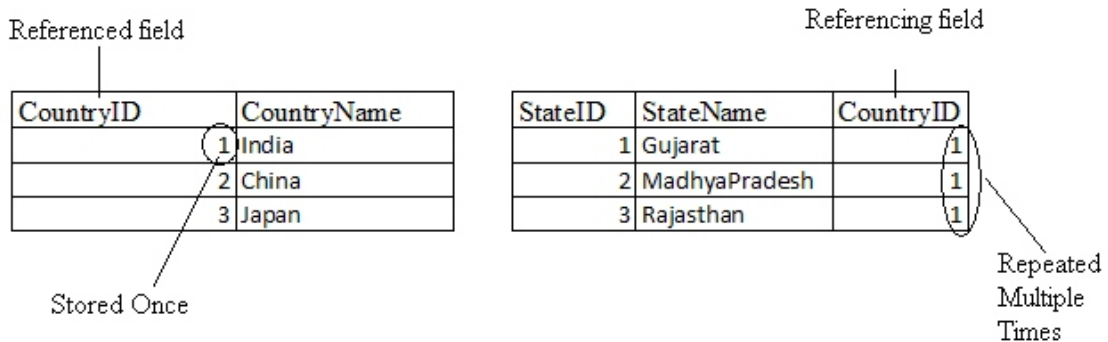
**Figure 10.9 : Relationships dialog box**

- To create a relationship, just use ‘drag and drop’ operation. Click on the CountryID field (primary key) of the Country table.
- Drag the CountryID field of the Country table and drop it on the CountryID field (foreign key) of the State table.
- A line connecting both the fields with labels *1* and *n* will be displayed as seen in figure 10.10. Notice that label text *1* is displayed on the primary key side and label text *n* is displayed on foreign key side. This indicates that the CountryID field is a primary key and will hold unique values, and each unique value stored in primary key field may be repeated *n* number of times in foreign key. This type of relationship where one value of referred table is associated with multiple values in referencing table is called *One-to-Many* relationship. Figure 10.11 shows some sample records stored in One-to-Many relationship. Types of relationships are also discussed in detail later in this chapter.



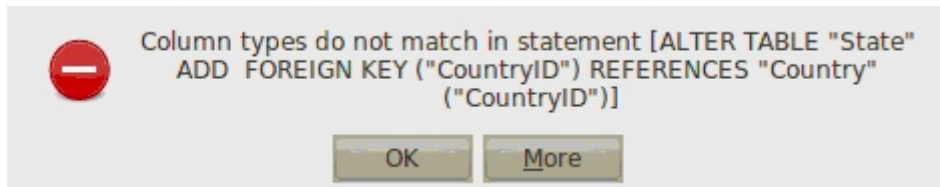
**Figure 10.10 : One-to-Many relationship between Country and State**

Figure 10.10 shows the relationship between the tables Country and State. Both the table shown in figure 10.10 can be moved to any desired location by clicking their title bar and then dragging it. Here the CountryID field of the Country table is known as *referenced field* while the CountryID field of the State table is known as *referencing field* (see figure 10.11).



**Figure 10.11: Records indicating One-to-Many Relationship**

**Note :** To create a relationship between two tables, the data types of *referencing field* and *referenced field* must be same. In case data type of both the fields do not match, error as shown in figure 10.12 would be displayed while creating relationship.



**Figure 10.12 : Column data type mismatch error in creating relationship**

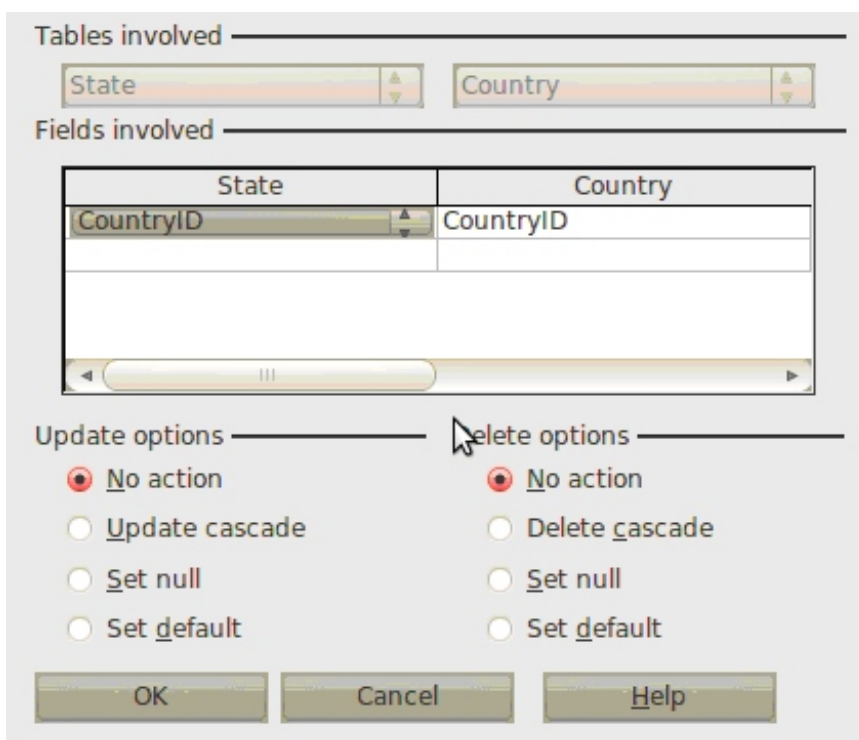
### Referential Integrity

Once the relationship between two tables is created, double-click the line depicting relationship. Relations dialog box as shown in figure 10.13 will be displayed. Till now, we are very clear with the statement that the Country table and the State table are related with each other. Thus a user

can enter data of only those country's details in the State table which have been already entered in the Country table. As seen in figure 10.11, first India was inserted in the Country table. Later three states of India were inserted in the State table. Record pertaining to India in the Country table can be considered as *master or parent* record, while records pertaining to India in the State table can be considered as *transaction or child* records.

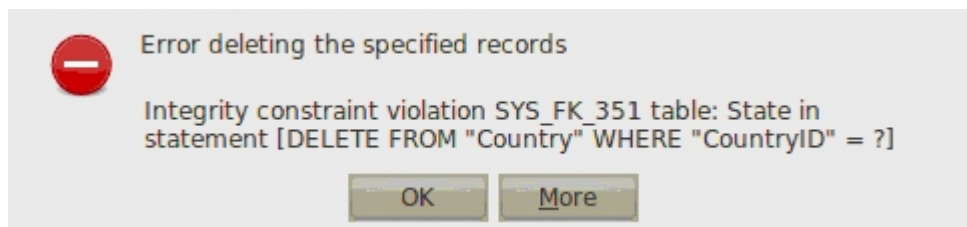
But what if now user deletes a parent record from the Country table? What about its related record in the other tables? Say for example, we have entered say ten records in the State table that have used CountryID as 1. What should be done with records of the Country table with CountryID as 1 in the State table, if the user decides to delete or update this record from the master table? There must not be any entry in the State table without a related record in the Country table. This concept is known as Referential Integrity. **Referential Integrity** principle can be stated as, *No unmatched foreign key values should exist in the database.*

The database designer shall choose and set one of the four options shown in figure 10.13 depending on the transaction requirement of the company. These options allow us to maintain referential integrity in database while performing an update or delete operation.



**Figure 10.13 : Enforcing referential integrity**

**No action:** This option states that a user should not be allowed to delete or update any record if its related record exists in some other table. Select *No Action* option and then try to delete the record containing India's details. Base will confirm whether you want to delete records or not and then will display message as shown in figure 10.14.

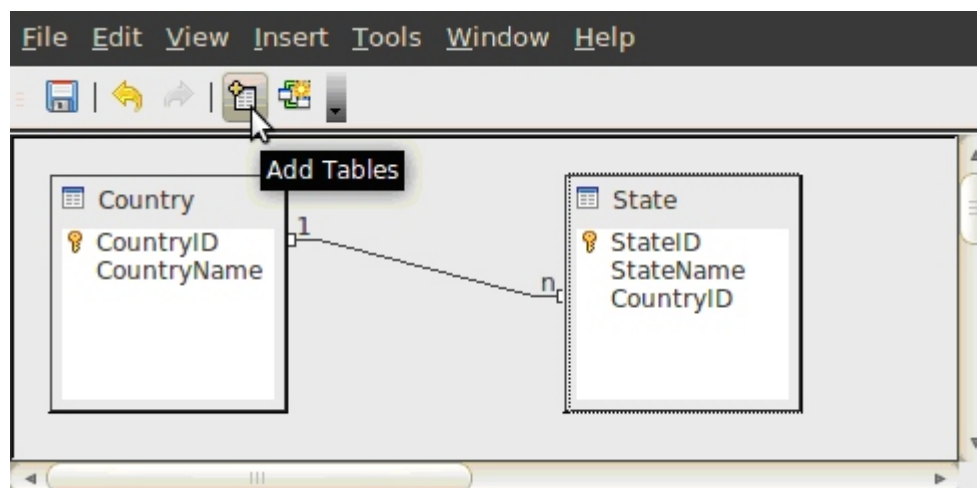


**Figure 10.14 : Enforcing Referential Integrity using No Action**

**Update cascade:** This option states that if user is allowed to delete or update referenced values, all the related records should be automatically deleted or updated.

**Set null:** This option states if user deletes or updates the referenced field, all the related records will hold null value in the related field.

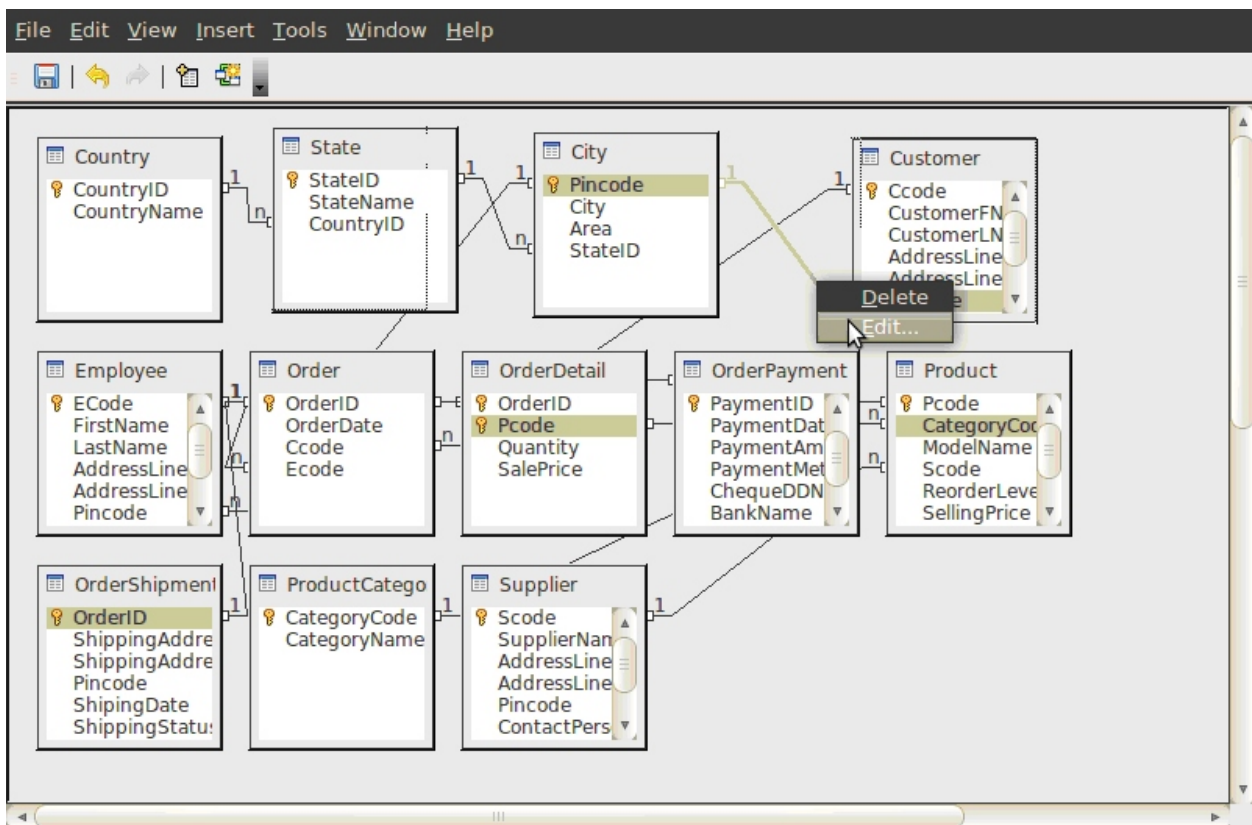
You can similarly create the relationship between the other tables of the Modern Electronic Store database. You can add other tables by clicking on *Add Tables* icon in tool bar as shown in figure 10.15. *Add Tables* dialog box as shown previously in figure 10.9 will be displayed.



**Figure 10.15 : Adding Tables in Relationship window**

Once you create the relationship between all tables of our sample database, relationship window will look similar to the one shown in figure 10.16.

Now as it is necessary to save the relationship we have recently established, we must save it by clicking on the Save button on the tool bar. In case we forget to save, then to remind us, an alert message will be shown on the screen. At times, changes made in relationships are not reflected in database after saving also. If such a thing happens then close the database and reopen it. You will find that modifications in relationships are implemented and visible.



**Figure 10.16 : Relationship of all tables in Database**

### Editing Relationships

We can delete or edit the relationship established between any two tables. For that, we just have to open the relationship screen and then click on the desired relationship line drawn between two tables. By doing so, the line will get thickened. After this if we right click, a popup menu will be visible as shown in figure 10.16.

Here, we can select the necessary option depending upon our requirement. If we select the Edit option from the popup menu then a Relationship dialog box will be shown where we can edit the relationship attributes as per our requirement. Alternatively we may select Delete option and remove the relationship. So now we know how to create, edit or delete various types of relationships between any two tables of the database. Create relationships between all the tables if not already created.

Now, let us discuss types of relationships in detail. Conceptually there can be three major types of relationship between any tables of the database.

- One to One
- One to Many
- Many to Many

Type of relationship among the data of two tables is defined based on how many corresponding records can be there in second table corresponding to the first table. Let us understand each of them properly.

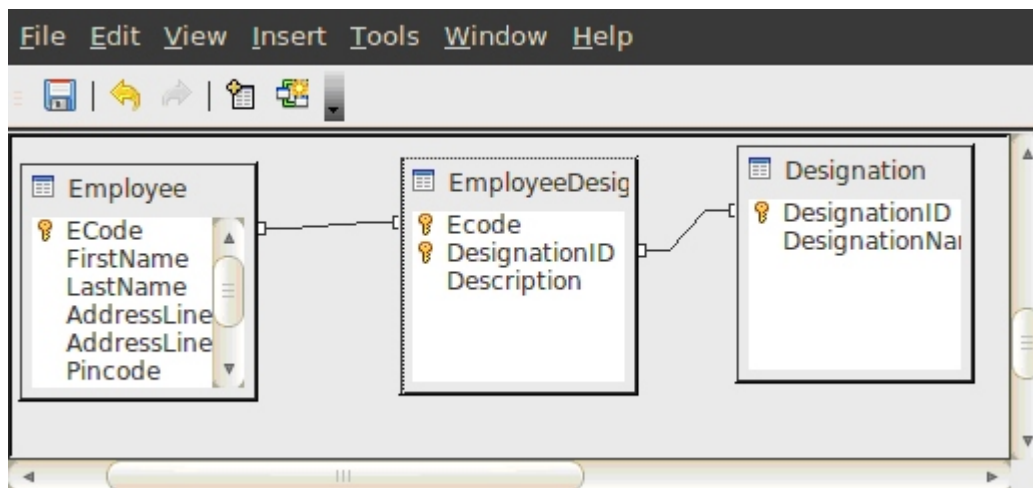
**One to One Relationship :** As its name suggests, One to One Relationship indicates that one specific record of a particular table has one and only one corresponding record in the other table of the database. Do not get surprised, as it's not new to us and very common in our day-to-day life examples. Think of a theatre. How many people can seat in a chair? Naturally, One. Then how many seats can be occupied by one person? Definitely, One. So, we can say that the relationship between Chair and Person is One-to-One Relationship. You can think of relationship between your class and class teacher or library and library card issued to a student. You will definitely agree that, any class in your school will have one and only one class teacher. At the same time any teacher is assigned one and only one class to look after. So we can say that, the Class and Class Teacher is an example of One-to-One relationship.

Observe relationships created in our sample database. The field OrderID in the Order table is related to the field OrderID in the OrderShipment table. Now only one record corresponding to each order for example, 1 would be found in OrderShipment table. It indicates that One-to-One relationship exists between the Order and OrderShipment tables. You may think then why we have not added the fields of OrderShipment to the Order table. In ModernElectronicDatabase case, the OrderShipment details are not to be populated for every order. The OrderShipment details are to be recorded only when customer places order online or expects the delivery of products at home or some other address when he has physically walked into the store and done the purchasing. Thus if the Order table is not decomposed, Null values will be populated in all the OrderShipment fields. Hence, it is better to decompose tables and maintain One-to-One relationship in such cases.

**One-to-Many Relationship:** The most common type of relationship between the tables is One-to-Many relationship. In this type of relationship one specific record of a particular table may have many corresponding records in the other related table of the database. One customer can purchase multiple products. Similarly, one city belongs to single state, but one state can have multiple cities. Imagine a class teacher scenario in your school, there can be one and only one class teacher to any class. But any class teacher definitely have many student enrolled with him. That's why the relationship between Class Teacher and Student is One-to-Many.

**Many-to-Many Relationship:** The Many-to-Many relationship occurs in the case of some specific record, which has to be stored more than once in both the tables. In our example of ModernElectronicStore database, we have maintained information of employees. Till now we have assumed that an employee will hold a single designation. At times in an organization an employee may hold multiple designations. Thus in that case we can state that an employee may have multiple designations and a single designation may belong to multiple employees. Separate table that stores employee designation has to be maintained in that case. Thus there exists Many-to-Many relationship between these two entities. Figure 10.17 shows a sample many-to-many relationship scenario.





**Figure 10.17 : Many-to-Many relationship between Employee and Designation**

But you can see in figure 10.17, we don't have any direct relationship between the tables Employee and Designation. This is because in database Many-to-Many relationships are split into two One-to-Many relationships by creating a third table. In our example we have created a new table named EmployeeDesig, which has One-to-Many relationship with the Employee table and Designation table both. Thus EmployeeDesig table acts as a junction table between the Employee and the Designation table.

Till now we have created tables, decided primary key fields and foreign key fields and related tables with each other. Now let us discuss about field properties.

### Field Properties

As discussed in the earlier chapter, before working with the data we have to create a well defined structure of tables. To define a field for any table we have to perform the following steps:

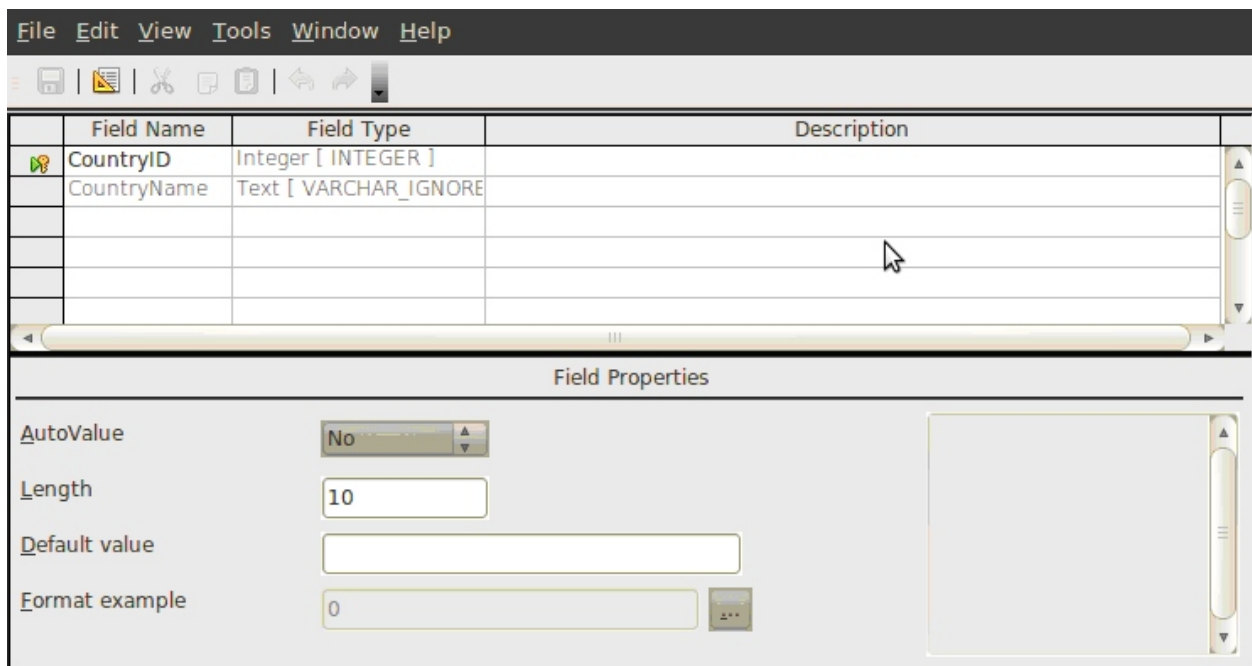
- Specify the *Field Name*
- Choose a proper *Data type* for the field
- Write *Description* of the field. (Optional)

In the field of information technology, the *data* is the raw material used to produce finished goods known as *Information*. It is a universal truth that 'We can't expect getting best without giving better'. In other words, we can say that the quality of the output obviously depends upon the quality of input. If we can control and restrict the values being stored at the time of data entry then, it can be the best practice to control the input data, which can generate the correct information.

What if a user leaves the Name field of the Customer table blank? Even well experienced data entry operators many times enter incorrect data by mistake. To prevent such mistakes, Base provides a special facility called '*Field Level Validation*'. It is nothing but putting some possible rules and checkpoints for data to be entered for each field. This further restricts the values entered in a particular field. Data validation is the important aspect of data processing. You must note that, if the data entered in various tables is correct and accurate then and then only the database will become a valuable asset to your organization.

Base has very important and interesting feature called *Field Properties*. **Field properties** determine how the values in the field are stored and displayed. Each type of field has a particular set of properties. For example, in what format will the user enter the data (dd-mm-yy or mm-dd-yy) in the OrderDate field, how should it be displayed while printing or viewing the data and what if user enters invalid date like 12/14/12 for a date format dd-mm-yy.

Field Properties can be decided at the time of designing the structure of the table. Select the field in Table Design Window, Base provides a *Field Properties* box as shown in figure 10.18 with appropriate default values according to the corresponding selected data type.



**Figure 10.18 : Field Property dialog box**

The various *Field Properties* work as field level validation for the table. *Field Properties* displayed on the screen may vary depending upon the data type that we have selected for the particular field. Let us discuss some of the common field properties in detail:

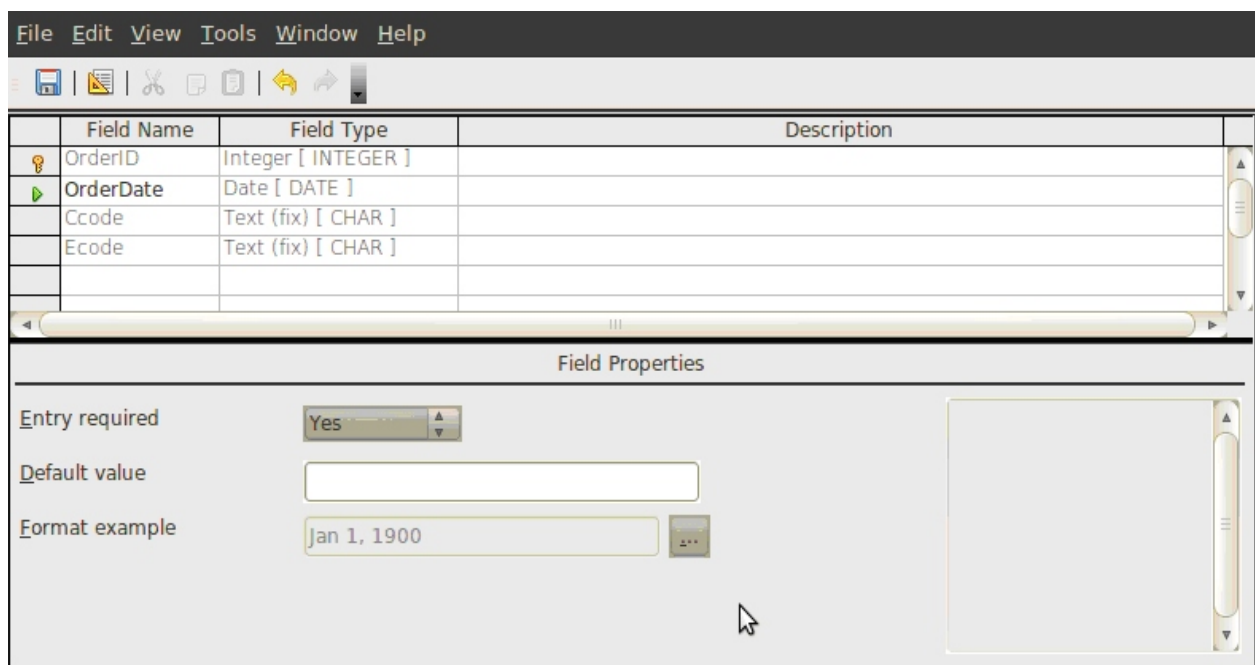
**AutoValue:** As can be seen in figure 10.18, *AutoValue* property has drop down box (combo box) next to it. Click on the drop down box and you will find two values, “Yes” and “No” within it. If the value of an *AutoValue* is set to “Yes” for a numeric field, Base can automatically increment the value for each new record. This field is particularly used for numeric fields designated as primary key. For example, the Country table consists of a field CountryID as primary key. The data type of the CountryID field is *integer*. The values expected for this field are 1, 2, 3... and so on. Thus *AutoValue* can be set to “Yes” in this case.

**Entry required:** Out of all the fields in the database there may be some important field, which

must not be left empty. Now, it may happen that the data entry operator forgets to enter the required value in important fields. How about, if we do not allow the user to leave such important field as blank at the time of data entry itself? We can achieve this requirement by setting the field property *Entry required*. The setting of this property can either be ‘Yes’ or ‘No’ (see figure 10.19).

This property determines whether the user must enter a value for a particular field to complete a record entry. We should set this property to ‘Yes’ if we do not want to allow the user to skip entry in any important field of our table. For example, we must set this property type for the fields that store value related to name of a person.

**Note :** We do not need to set the value of *Entry required* property to *Yes* for the field declared as *primary key*.

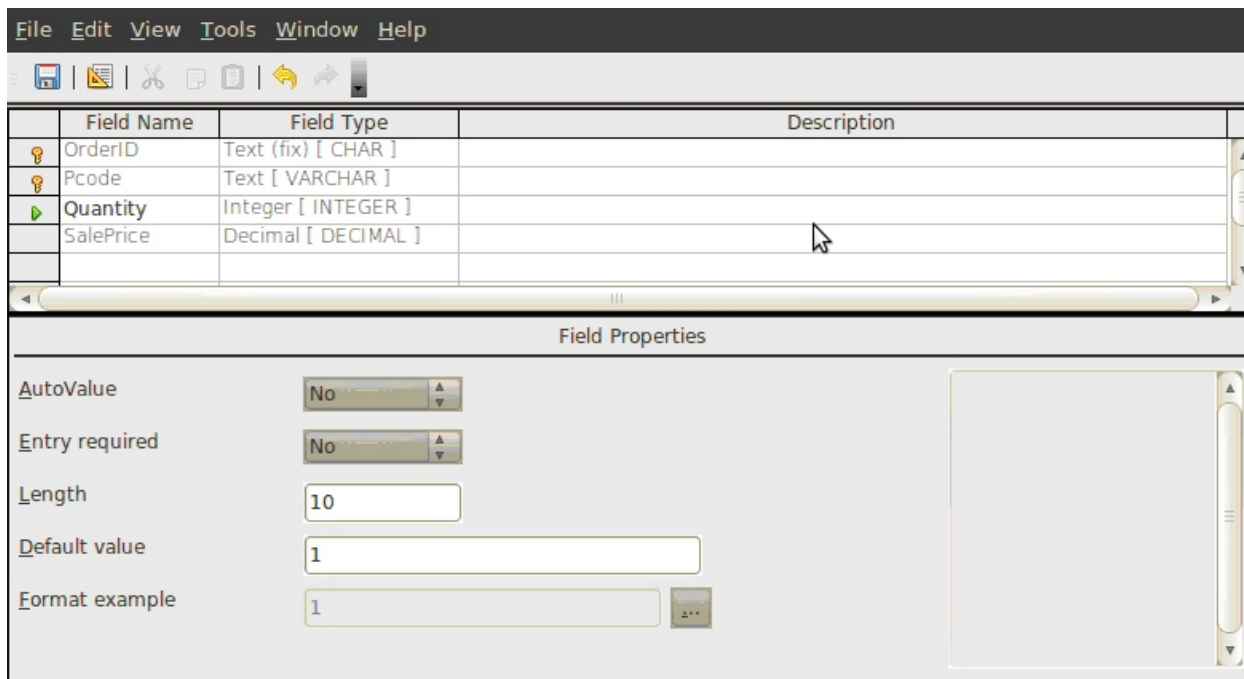


**Figure 10.19 : Entry required Property set to Yes**

**Length:** Whenever we select the *Text* data type for a particular field, this property automatically takes some predefined value e.g. 50 for *Text* data type. We can specify field size as per our requirement. Base automatically assigns some predefined field size to the various *Numeric*, *Date/Time*, *Yes/No* and *Memo* data types. That’s why in such cases this property will be disabled on the screen.

**Default value:** Many times it may happen, that we want to store some predefined default value automatically in a particular field at the time of entering a new record in the table. Suppose that, in the *OrderDetail* table we have a field named *Quantity*. Now, we may want to store value ‘1’

as a default value for this field. Select the Quantity field, type '1' in the text box next the *Default value* label as shown in figure 10.20. Once we set this property for a field, the specified default value will automatically be displayed when we add a new record to the table. A user can change the value if required at the time of data entry.



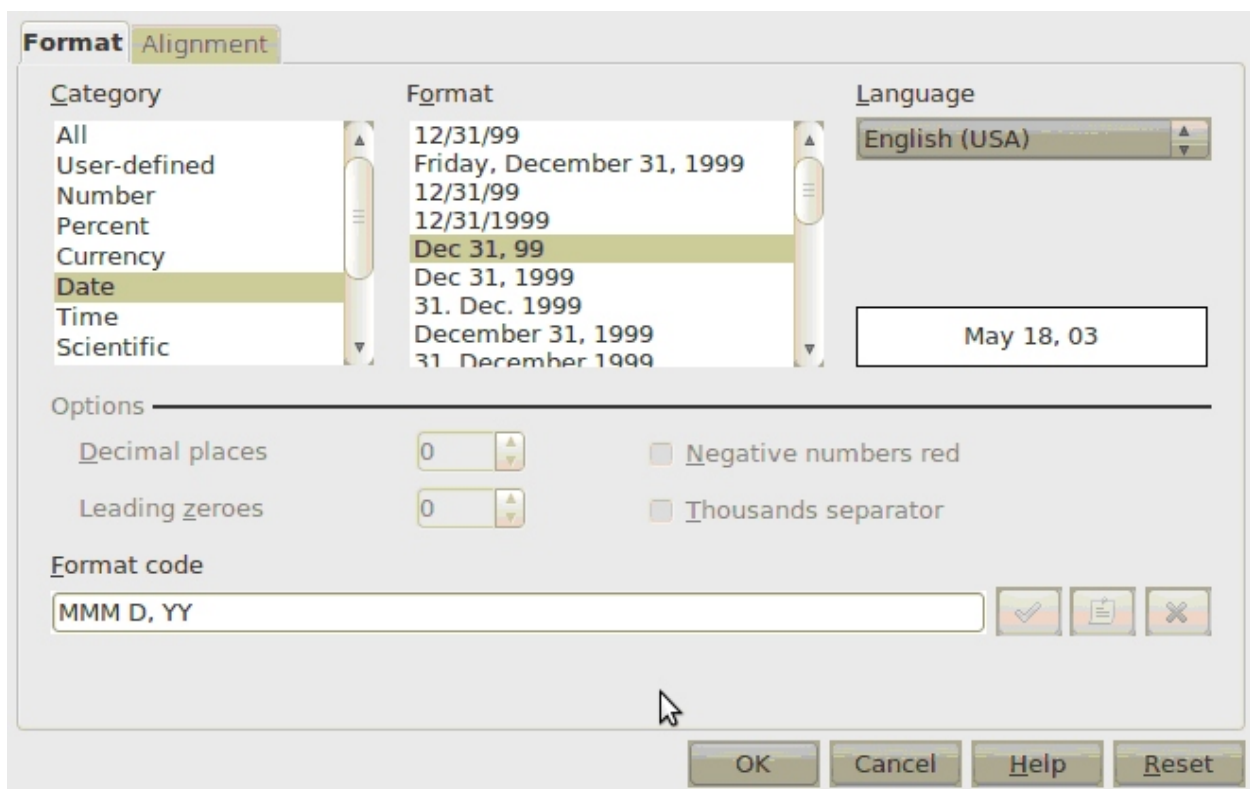
**Figure 10.20 : Specifying Default value for a field**

**Format:** This property specifies the format for displaying and printing the data that we enter in the field. It has no effect on the way the data is stored and it doesn't check for invalid entries. The Format property uses different settings for different data types. Base provides some predefined formats for *Number*, *Date/Time*, and *Yes/No* data types.

For example, to change format of the field OrderDate in the Order table,

- Open the Order table in Design view.
- Select the OrderDate field and click on button at the end of Format example label in Field Properties pane. A Field Format dialog box as shown in figure 10.21 will open.
- Select Date option from the list under the heading Category.
- Select Dec 31, 1999 option from the list under the heading Format.
- Click on OK button.

Now open the table and enter valid date in any format. You will find that Base automatically converts all the entry in this field in the format that shows first three characters of month, a space then two digit date and two digit year values separated by a comma.



**Figure 10.21 : Predefined formats**

### Summary

In this chapter we have discussed the operations that can be performed on the table. We learnt how to create relationship between the tables and types of relationship that can exist between the tables. Finally we learnt how to control data entry and restrict the user from entering wrong data in the table using Base feature called Field Properties. We need to populate designed database for Modern Electronic Database with meaningful records and in the next chapter we will see how the inserted data helps us generate information using Queries.

### EXERCISE

1. Explain how we can perform insertion operation in a table.
2. What do you mean by relations in Base?
3. Explain types of relationships giving suitable example.
4. Explain the concept of foreign key giving suitable example.
5. What do we mean by Composite primary key? Explain giving suitable example.
6. Explain data redundancy giving suitable example.
7. State referential integrity rule.
8. What does required field property signify?
9. Explain the importance of Default value in database.

10. Explain Autovalue property.
11. What is use of Entry required property?
12. What is the use of Format property?
13. Choose the most appropriate option from those given below :
- (1) Which of the following operations cannot be performed on a record in database?
    - (a) Insert
    - (b) Delete
    - (c) Hide
    - (d) Update
  - (2) Which of the following represents maximum levels of sorting provided by Base Wizard?
    - (a) 3
    - (b) 4
    - (c) 5
    - (d) 6
  - (3) Which of the following refers to data redundancy?
    - (a) Deletion of data
    - (b) Repetition of data
    - (c) Decomposition of tables
    - (d) Relationships of tables
  - (4) Given a relation, TeacherSubject (Tcode, Scode, Standard), which of the following fields can be an appropriate foreign key field?
    - (a) Tcode
    - (b) Scode
    - (c) Tcode,Standard
    - (d) Tcode,Scode
  - (5) Which of the following statement is incorrect?
    - (a) Every table must have a primary key.
    - (b) Primary key can be combination of fields.
    - (c) Data type of Primary key and Foreign key should be same.
    - (d) Names of Primary key and Foreign key should be same.
  - (6) Which of the following type of relationship exists between Student and Teacher in a school?
    - (a) One to One
    - (b) One to Many
    - (c) Many to Many
    - (d) No relationship exists
  - (7) Which of the following type of relationship exists between Student and Class Teacher in a school?
    - (a) One to One
    - (b) One to Many
    - (c) Many to Many
    - (d) No relationship exists
  - (8) Which of the following options are possible to implement referential integrity?
    - (a) No Action
    - (b) Set Null
    - (c) Delete table
    - (d) Update Cascade
  - (9) AutoValue property can be set for field with which of the following datatype?
    - (a) Text
    - (b) Image
    - (c) Integer
    - (d) Boolean

- (10) Which of the following statement is true for Default field property in Base?
- (a) Default value can be numeric value only.
  - (b) Default value once set can be changed later.
  - (c) Default value cannot be greater than 500.
  - (d) Default value should be greater than value set in Length.
- (11) Which of the following property is equivalent to NOT NULL?
- (a) Length
  - (b) Default
  - (c) Required
  - (d) Format

**LABORATORY EXERCISES**

1. The structure of the table is given in the form of Tablename(Attribute1, Attribute2,.....). Perform the following five operations for each database using Base.
- (a) Create tables with the given names and attributes.
  - (b) Decide appropriate data type for each attribute
  - (c) Identify primary key and foreign key in each of the database.
  - (c) Establish relationships amongst the tables created within a database.
  - (e) Insert at least five appropriate records in each table.

A.	Movie(MovieId, MovieName, DateOfRelease) Screen(ScreenId, Location, MaxCapacity) Current(MovieId, ScreenId, DateOfArrival, DateOfClosure)
B.	Customer(CustomerID, CustomerName, Address, City, BirthDate, ContactNo) Account (AccountNo, CustomerId, AccountType, AccountBalance) Transaction(TransId, AccNo, TransDate, TransType, TransAmount)
C.	Book(BookId, BookTitle, Description, BookAuthor, Status) Student(StudId, StudName, Address,City, BirthDate,ContactNo) Book_Issue(TransId, BookId, StudId, IssueDate, ReturnDate)
D.	Employee(EmpId, EmpName, Address, BirthDate, ContactNo, ManagerId) Department(DeptId, DeptName) Employee_Department(EmpId, DeptId, Salary)
E.	Flight(FlightId, CompanyName, FlightFrom, FlightTo, FlightFare, Capacity) Passenger(PassengerId, Name, Address, City, BirthDate, Gender, ContactNo) Flight_Scheduled(Transid, FlightId, DepartureDate) Flight_Passenger(Transid, PassengerId)

F.	Train(TrainId, TrainName, TrainFrom, TrainTo, DepartureTime, ArrivalTime) Train_fare(Trainid, Class, Fare) Passenger(PassengerId, Name, Address, City, BirthDate, ContactNo, Email_id) Train_Passenger(TrainId, PassengerId)
G.	Vehicle(VehicleId, Name, Type, Price, Description) Customer(CustomerId, CustomerName, Address, BirthDate, ContactNo) Vehicle_Customer(VehicleId, CustomerId, PurchaseDate, DeliveryDate)
H.	Product(ProductId, ProductName, Quantity, ProductPrice) Salesman(SCode, SName, SAddress, BirthDate, ContactNo) SalesOrder(SCode, ProductId, QtySold)
I.	Customer(CustomerId, CustName, Gender, CustAddr, CustCity, EmailID, ContactNo) Magazine(MagazineId, MagazineName, UnitRate, Publication) Subscription(CustomerId, MagazineId, StartDate, EndDate)
J.	Employee(EmpCode, EmpName, EmpAddress, EmpCity, EmpSalary, EmpJobName) Project(ProjCode, ProjName, StartDate, ProjPrice) WorksFor(ProjCode, EmpCode, HoursWorked)

2. Identify primary key and foreign key from the sample tables shown in figure 10.22 and 10.23.

<b>Student</b>	
<b>Field Name</b>	<b>Description</b>
Gmo	General Register Number
Firstname	Name of the Student
Surname	Surname of the Student
Address	Address of the Student
City	City
Pincode	Pincode
Birthdate	Date of Birth
Gender	Male or Female
Standard	Studying in which standard
Join_date	Date of Joining School
Leaving_date	Date of Leaving School

<b>Teacher</b>	
<b>Field Name</b>	<b>Description</b>
Firstname	Name of the Teacher
Surname	Surname of the Teacher
Address	Address of the Teacher
City	City
Pincode	Pincode
Phone_no	Phone number of teacher
Email_id	E-mail id of teacher
Mobile_no	Mobile number of teacher

<b>Subject</b>	
<b>Field Name</b>	<b>Description</b>
Sub_Name	Name of the Subject
Details	Description of the subject

**Figure 10.22 : Master tables of Student Management System**



<b>Standard_Subj</b> : To store details about subjects taught in each standard		
Field Name	Data Type	Description
Standard	Number	Which Standard
Scode	Text	Which Subject

<b>Attendance</b> :To store daily attendance of the Students		
Field Name	Data Type	Description
Grno	Number	General Register of a Student
Date	Date/Time	On which Date
Pr_ab	Yes/No	Present or Absent

<b>Fees</b> :To store details of Fee amount paid by students		
Field Name	Data Type	Description
Grno	Number	General Register Number
Date	Date/Time	On which date
Tuition_fee	Currency	Tuition fee paid by the student
Term_fee	Currency	Term fee paid by the student

<b>Cultural</b> :To keep record of Interest of students in Cultural Activities		
Field Name	Data Type	Description
Grno	Number	General Register Number
Interest	Text	Interested in which activity
Achievement	Memo	Any achievements?

<b>Subj_Teacher</b> : To store details about which teacher is teaching the subject in each standard		
Field Name	Data Type	Description
Year	Number	Current Year
Standard	Number	In which Standard
Scode	Text	Subject Code
Tcode	Text	Teacher Code

<b>Result</b> : To store result details of each student		
Field Name	Data Type	Description
Grno	Number	General Register No.
Scode	Text	Of which Subject
Month	Text	In which Month
Year	Number	Which Year
Term	Number	Name of the Term
Marks	Number	Marks scored

<b>Sports</b> :To keep record of Interest of students in Sports		
Field Name	Data Type	Description
Grno	Number	General Register No.
Interest	Text	Interested in which game
Achievement	Memo	Any achievements?

**Figure 10.23 : Transaction tables of Student Management System**

