

---

## Introduction

---

Paradox is a *database management* system, which is to say it works with data stored in databases. Most people understand the role of word processing and spreadsheet programs, and understand the use of a database management program for simple list keeping, such as a list of customers used to create mailing lists. Fewer people, however, understand the role of *relational* databases in business and how a relational database management program works with those databases.

The word relational has one meaning to a database scientist, and a different, less well-defined meaning in common usage, as database software vendors have found it advantageous to claim their products are relational. In common usage, a relational database management system has the capability of working with information stored in two or more *tables* at a time. (In relational database language, tables or relations are what are commonly referred to as databases.) Paradox is a very good relational database management system that you can use to create quite sophisticated database management systems.

Paradox contains several key areas that you'll want to master:

*Table Creation* is the process of defining your data tables to Paradox, including how to divide data across the appropriate number of tables, and defining the proper column types to hold the data.

*Data Entry and Editing* is the process of adding data to the tables that you've defined.

*Queries* represent a broad range of skills in Paradox that let you manipulate your databases in a variety of ways, such as asking questions about the data, performing calculations and summaries, joining data from different tables, updating data, and deleting data.

*Reports* are the way that you print data from your tables.

*PAL (Paradox Application Language)* is a programming language within Paradox that lets you create applications. Not everyone needs to use PAL in order to use Paradox effectively, and those who aspire to use PAL need a thorough grounding in the topics mentioned above.

---

## Paradox Function Keys

---

Paradox makes use of function keys for many commands and operations. Here's a table of the function keys and their meanings:

Key	Meaning
F1 Help	The <i>Help</i> key. Displays a help screen containing information on using Paradox.
F2 Do_It!	The <i>Do_It!</i> key, used to end or perform many operations. For example, when you finish filling out a query form, you press F2 to perform the query.
F3 Up Image	The <i>Up Image</i> key, which moves up one image in the workspace.
Alt-F3 Instant Script	The <i>Instant Script</i> key, which begins recording an instant script. This key also ends recording an instant script.

<b>Key</b>	<b>Meaning</b>
F4 Down Image	The <i>Down Image</i> key, which moves down one image in the workspace.
Alt-F4 Instant Script Play	The <i>Instant Script Play</i> key, which plays back the instant script.
F5 Example	The <i>Example</i> key, used to enter an example element in a query.
Alt-F5 Field View	The <i>Field View</i> key, used to enter a field when editing or during data entry.
F6 Checkmark	The <i>Checkmark</i> key, used to include a field in the result of a query.
Alt-F6 Check Plus	The <i>Check Plus</i> key, used to include all records, including duplicates, in the result of a query.
Ctl-F6	The <i>Check Descending</i> key, used to include a field in the result of a query, but to sort the result in descending order.
F7 Form Toggle	The <i>Form Toggle</i> key, used to switch between the table view and form view.
Alt-F7 Instant Report	The <i>Instant Report</i> key, which sends a report of the current table to the printer. Make sure your printer is ready before you press this key.
F8 Clear Image	The <i>Clear Image</i> key, which removes the image of the current table from the workspace.
Alt-F8 Clear All	The <i>Clear All</i> key, which removes all images from the workspace.
F9 Edit	The <i>Edit</i> key, which enters edit mode so you can make changes to a table.
Alt-F9 CoEdit	The <i>CoEdit</i> key, which enters editing mode so that you, and others using the same table on a network, can make changes to the table.
F10 Menu	The <i>Menu</i> key, which displays the Paradox menu so you can issue commands from the menu.
Alt-F10 PAL Menu	The <i>Pal Menu</i> key, which displays a special menu of PAL (Paradox Application Language) commands.

## Paradox Movement Keys

The meaning of each key on the cursor keypad varies, depending on which mode you're working in.

<b>Key</b>	<b>Meaning</b>
Up arrow, Down arrow	In table view, moves up or down one record or row. In form view, up or down one field.
Left arrow, Right arrow	In table view, moves left or right one field. In form view, moves to previous or next field.
Home, End	Moves to the first or last record in the table.
Page Up, Page Down	In table view, moves up or down one screen of records. In form view,

Key	Meaning
	moves up or down one page or record.
Control-Left arrow Control-Right arrow	In table view, moves left or right one screen of fields.
Control-Home, Control-End	Moves to the first or last field in the current record.
Control-Page Up Control-Page Down	In form view, moves to the same field of the previous or next record.
Tab, Shift-Tab	Moves to previous or next field.
Insert	Inserts a record in the table. When in field view, turns insert mode on and off.
Delete	Deletes the current record from the table.

## Other Keys

Key	Meaning
Backspace	Erases characters to the left of the cursor.
Control-Backspace	Erases current field.
Control-D	The <i>Ditto</i> key. Enters the value in the corresponding field from the previous record.
Control-F	The <i>Field View</i> key. Moves to field view mode of editing (same as Alt-F5).
Control-R	The <i>Rotate</i> key. Rotates fields to the right of the cursor.
Control-Z	The <i>Zoom</i> key. Finds values in a field.
Alt-Z	The <i>Zoom Next</i> key. Finds the next matching value in a field after first using the <i>Zoom</i> key.

## Paradox Menus

To perform many commands, you make a selection from a menu. Paradox menus work nearly the same as Lotus 1-2-3's menus, so if you've used 1-2-3, you know how to make menu commands in Paradox.

To select a command from a Paradox menu, follow these steps:

First, press F10, the Menu key. Then follow either of these steps:

Use Left arrow and Right arrow to highlight the command you want to perform. Once the command is highlighted, press the Enter key.

Type the first character of the command you want to perform. If you want to do the Report command, for example, type R. Don't press the

---

Enter key after typing the character.

You can use either method as you like. The first method is good for the new Paradox user, because it familiarizes you with the menus. The second method is the way most people use Paradox after they've had some experience with the program.

When using menu commands that request the name of a Paradox object, such as a table name or script name, you can of course type the name of the table or script directly when Paradox asks for it. A second method is to follow the instructions on the screen to press the Enter key. After pressing Enter, Paradox presents a list of tables or scripts. Use the arrow keys to highlight the one you want, and then press Enter to select it.

A second shortcut is handy when the list of tables is long. To use this method, follow these steps:

- 1) Issue the *View* command and press the Enter key to get the list of tables.
- 2) Press the first letter of the name of the table you want. Paradox then presents just the tables that start with that letter.
- 3) Use the arrow keys to highlight the table name that you want and press the Enter key. If there was only one table starting with the letter you pressed, Paradox selects it automatically.

---

## Setting the Default Directory

---

To set the default directory (the drive and directory Paradox uses for storage), follow these steps:

- 1) Issue the *Tools More Directory* command.
- 2) Delete the existing directory name (*Control-Backspace* is a shortcut).
- 3) Type the new drive and directory that you want to use and press the Enter key.
- 4) Select *OK* from the next menu to confirm your action.

---

## Tables

---

Paradox works with data stored in tables. A table is much as you'd expect—rows and columns of data. In a database program a row of data is sometimes called a *record*, and a column of data is called a *field*.

Paradox tables have structures that you define. The key elements of the structure for each field are the field's name and data type. Field names may be up to 25 characters long. They may contain spaces, but can't start with a space. You can have up to 255 fields in each table. Paradox field types are as follows:

Field Type	Description
A (Alphanumeric)	May contain any character that you can type from the keyboard. Alphanumeric fields have a length that you specify. The length represents the longest entry you'll be able to make, and determines the amount of disk space the field requires.

Field Type	Description
	The length may be from one to 255 characters.
N (Number)	Number fields may contain numbers of up to 15 significant digits. Paradox lets you format the numbers' appearance in different ways.
\$ (Dollar)	Dollar fields are like number fields, but Paradox places commas at the thousands points, displays the number with two decimal places, and shows negative numbers in parentheses. These fields do not display with dollar signs, in spite of the name.
D (Date)	Date fields contain dates from January 1, 100 to December 31, 9999. You enter dates in the form MM/DD/YY or DD-Mon-YY (examples for January 15, 1988: 1/15/88 and 15-Jan-88). Paradox lets you display the dates in a number of formats.
S (Short Number)	Short number fields may contain only integers from -32,767 to 32,767. These fields are special and should not be used under normal conditions.

You may designate one or more fields as key fields. Paradox always maintains the table in key field order, so key fields can help eliminate sorting the table. The values in key fields must be unique. If you attempt to enter duplicate records in a key field, Paradox places the duplicate records in a special table called *Keyviol*. Additionally, queries or searches based on the key field will be much faster than queries based on non-key fields.

Create key fields as regular, but type an asterisk after the field type. For example, A10\* is an alphanumeric field of length 10 that is also a key field.

To create a new table, follow these steps:

- 1) First, make sure that Paradox is using the drive and directory that you want to create the table in. Use the *Tools More Directory* command (as described above) to change if necessary.
- 2) Issue the *Create* command from the menu.
- 3) Type a table name. Paradox table names are DOS file names and follow the normal rules for DOS file names. Press the Enter key after entering the file name.

You need to follow the usual DOS file naming rules, which means that you can use up to eight characters composed of the alphabet letters (upper and lower case don't matter here), the numeric digits, and the \$ & # @ ! % ' ~ ( ) - \_ { } characters. Don't use spaces in a file name. Paradox automatically adds the extension *.db* to whatever file name you supply.

- 4) Enter field names and field types.
- 5) Press *F2 Do\_It!* or select *Do\_It!* from the menu when finished.

## Restructuring Tables

---

You may need to restructure a table when you forget to include fields when you first create a table, or when you wish to delete, rename, or change the type of existing fields. Follow these steps to restructure a table:

- 1) Issue the *Modify Restructure* command.
- 2) Specify the name of the table to restructure and press the Enter key.
- 3) Make changes to the table structure.
- 4) Press *F2 Do\_It!* or select *Do\_It!* from the menu when finished. It may take Paradox some time to complete the restructure if the table has many records.

To delete a field, move to the field and press the Delete key. To add a field, either move to the bottom of the field list and type the field name and field type, or press the Insert key to insert a blank row above the current row for a field. Use the regular editing techniques to change a field's name or field type. To move a field, use the Insert key to add a new row at the new position for the field. Type the field's name at the new position and press the Enter key. Paradox will then move the field to the new position.

## Problems During Restructure

---

If you delete fields during a restructure, Paradox will ask you to confirm the deletion of each field after you press *F2 Do\_It!*. Select *Delete* from the menu to go ahead and delete the field from the table, or select *Oops!* from the menu to return to the restructure and add the field back to the table.

When changing field types, Paradox usually does a very good job converting one field type to another. There may be times, however, when Paradox needs help performing a conversion. For example, suppose you currently have an alphanumeric field that contains numeric values. You want to convert this field to a numeric field so that you can perform calculations. By editing the *A10* (or whatever it is) to *N*, you can make the change. But suppose that one of the values in the converted field is *1243W3*. This can't be converted to a numeric field because of the *W*. Paradox, in this case, creates a special temporary table called *Problems* that will contain this record, plus additional records that Paradox can't convert. You should edit the values in the *Problems* table and then add them back to the original table.

The *Problems* table may arise in another situation. Suppose a field is currently *A25* and you shorten it to *A20*. There is, in this case, a potential for data loss. After you press *F2 Do\_It!* to complete the restructure, Paradox will ask you what to do about possible data loss in this field. If you select *Oops!* from the menu, Paradox returns to the field table so you can make additional changes. If you select *Trimming*, Paradox discards the excess characters from the original field that won't fit in the new length. If you select *No-Trimming*, Paradox places values that are too long for the new length of the field in the *Problems* table. You can then edit the records in the *Problems* table and add them to the original table.

Paradox requires that values in key fields (or a combination of fields that make a key) be unique. If you make fields into key fields, and these key fields contain duplicate values, Paradox detects a key violation. Paradox places duplicate records in a table called *Keyviol*. You should edit these records and then add them to the original table.

## Viewing Tables

---

To view Paradox data, use the *View* command from the menu. Type the table name to view, or press the Enter key to get a list of tables.

Once you're viewing the table, use the arrow keys to move about. You can press *F7 Form Toggle* to move between the table and form views. To view additional tables of data, issue the *View* command again. To move between several tables being viewed at once, use *F3 Up Image* and *F4 Down Image*. To remove a table from the workspace, use *F8 Clear*. A shortcut for removing all tables from the workspace is *Control-F8 Clear All*.

When viewing tables, the tables are protected from changes. If you try to make changes, Paradox presents a message at the bottom of the screen notifying you of this protection. To make changes, press *F9 Edit* or *Alt-F9 CoEdit*.

## Image Settings for Tables

---

Paradox lets you make several types of changes to a table's image on the workspace. These changes do not affect the data in the table, or the structure of the table, but only the appearance of the data.

## KeepSet

---

Changes that you make with the Image set of commands are not retained permanently unless you perform an extra step. This means that you might spend some time adjusting the image of a table, but if you fail to perform this extra step, Paradox will not retain the settings you made.

The extra step to perform is to issue the *Image KeepSet* command. Paradox then records the settings for the table, and the next time you view the table, Paradox will use your settings.

## TableSize

---

The first setting you might change is the table size. With this command, you can control how many rows of a table Paradox displays. To use it, follow these steps:

- 1) View the table that you want to modify.
- 2) Select the *Image TableSize* command.
- 3) Use the up and down arrow keys to adjust the number of rows.
- 4) Press the Enter key when finished.

## ColumnSize

---

Paradox lets you adjust the width of the individual columns of the table's image. You might do this to narrow the columns so that you can see more at a time. Follow these steps to adjust the width of a column:

- 1) View the table that you want to modify.
- 2) Select the *Image ColumnSize* command.
- 3) Use the left and right arrow keys to move into the column that you want to adjust. Press the Enter key when there.
- 4) Use the left and right arrow keys to adjust the width of the column. Press the Enter key when finished.

Remember to issue the *Image Keepset* command if you want to permanently retain these settings.

When making columns narrower, you may not be able to see all of your data. This does not affect the data stored in the table, just whether Paradox can display it on the screen. For alphanumeric and date columns, Paradox simply truncates the displayed data if it won't fit in the width of the column. For numeric fields, Paradox displays asterisks. In any case, you can move to the field and press *Control-F5 Field View* to move about the field and see the entire contents.

## Format

---

The *Image Format* command lets you adjust the formatted appearance of numbers and dates. You may format numbers as General, Fixed, Comma, or Scientific, with a number of decimal places that you desire. The Scientific format displays numbers in scientific notation. The Comma format displays numbers with inserted commas at the thousands points, with negative numbers in parenthesis. The General and Fixed formats don't use commas or parenthesis. The difference between the two is that Fixed format always displays the indicated number of decimal places, even though the decimals may be all zero. The General format would not show the all-zero decimals.

For dates, you can format as MM/DD/YY, DD-Mon-YY, or DD.MM.YY. Using February 15, 1989 as an example, dates look like this with these formats: 02/15/89, 02-Feb-89, and 15.02.89.

To format a field, follow these steps:

- 1) View the table that you want to modify.
  - 2) Select the *Image Format* command.
  - 3) Use the left and right arrow keys to move into the column that you want to format. Press the Enter key when there.
  - 4) Select the desired format from the menu.
  - 5) If formatting a numeric field, type the number of decimal places to show.
-

Remember to use *Image Keepset* to keep the settings permanent.

## Moving a Field

---

Through the *Image Move* command, you can change the placement of a field. You might do this to group commonly viewed fields at the beginning of the table. To move a field, follow these steps.

- 1) View the table that you want to modify.
- 2) Issue the *Image Move* command.
- 3) From the menu of field names that appears, select the field that you want to move.
- 4) Move the cursor to where you want to place the field. Paradox inserts the moving field in front of the field you select.

Remember to use *Image Keepset* to make the adjustment permanent.

## Editing Tables

---

As mentioned before, when viewing Paradox tables, you can't change them. To change tables, you must enter either edit or coedit mode. Then, you're free to make changes.

The difference between edit and coedit applies when accessing tables that are stored on a network for common access. When you edit a table, you place a lock on the table so that you have exclusive access to the table. When coediting, others can access and coedit the table at the same time you are.

To edit a table follow either of these methods:

Use the *View* command to view the table you want to edit. Then press *F9 Edit* or *Alt-F9 CoEdit*.

Issue either the *Modify Edit* or *Modify CoEdit* command. Select the table you want to edit.

With either method, you'll be editing the table. The advantage to the function key method is that often, you'll already be viewing the table that you want to edit. Additionally, if you are viewing several tables, pressing *F9* or *Alt-F9* enters editing mode for all tables.

When editing a table, move through the table with the regular movement keys. You can edit in table or form view, whichever is most convenient, and toggle between the two modes freely. You can also use *F3 Up Image* and *F4 Down Image* to move to other tables on the workspace.

To make changes to a field value, first move to the field. Then, follow these guidelines:

Key	Meaning
Backspace	Deletes characters from the end of the field so that you can type new characters.

Key	Meaning
Control-Backspace	Deletes all characters in the field. Useful for clearing a field for typing a new entry.
Alt-F5 <i>Field View</i> Control-F	Enters field view mode. When using field view, you can use the left and right arrow keys to move through the field. The Delete key deletes the highlighted character. Characters that you type are inserted in front of the highlighted character.
Control-D	The <i>Ditto</i> key, which duplicates the value from the field above. The current field must be blank before using this key, so use Control-Backspace to delete it if necessary.
Delete	Deletes the current record. Be careful with this, as with most programs, the Delete key deletes a single character, but in Paradox it deletes an entire record of data.
Insert	Inserts a blank record above the current record.
F2 <i>Do_It!</i>	Leaves editing mode, making your changes permanent.
Control-U	The <i>Undo</i> key, which reverses the last change you made.

When you're finished editing a table, press *F2 Do\_It!* or select the *Do\_It!* command from the menu.

## Undoing Editing Changes

---

While editing a table, Paradox keeps a log of the changes you're making, allowing you to reverse all or some of the changes you've made.

One type of undo is to discard all the changes you've made since you started editing the table. To do this, select *Cancel* from the menu after pressing *F10 Menu*. You'll have to confirm your intention to cancel the editing changes. If you select *Yes*, Paradox restores the table to the exact state it was in before you started editing. Even if you delete all the records during the edit, they'll all come back if you cancel the edit.

After pressing *F2 Do\_It!* or selecting the *Do\_It!* command from the menu to complete an editing session, your changes are permanent and can't be undone.

The second type of undo is to press *Control-U Undo* or to select *Undo* from the menu. Paradox then undoes the last change that you made. For example, if you mistakenly deleted a record, Paradox will restore it. If you made a change to the wrong field, Paradox restores the field's original value. You can undo as many edits as you want in reverse order, but you must undo them in order. You can't undo the next-to-last edit without also undoing the last edit.

## Zooming to Locate Data

When editing data, you often need to find the particular record that you need to edit. The record numbers at the left of the table view don't often help, because the record number doesn't correspond to the content of the data records. The two zoom keys can help. These keys simply move to a particular record. After moving to the record of interest, you can view it or edit it as needed.

To search for a value, follow these steps:

- 1) Use the arrow keys to move to the field that you want to search through. If you want to find someone by their last name, move to the field containing the last name.
- 2) Press *Control-Z Zoom*.
- 3) Type the value that you're looking for and press the Enter key.

If the search is successful, Paradox moves to the record. Otherwise, Paradox displays a not found message.

If the record Paradox found is not the one you want, you can press *Alt-Z Zoom Next*. Paradox moves to the next record that matches the value you're looking for.

To look for a new value, start with *Control-Z Zoom* again.

Paradox also includes pattern operators that help match data when zooming to records. The most useful pattern operator is .. (two periods), which match any number of characters. Here's a few examples of the use of this pattern operator:

John..	Matches any name beginning with John: Johnny Carson, John Wilson.
..John..	Matches any name with John somewhere in it: Steve Johnson, Jack Johns, Johnny Carson.
..John	Matches any name ending with John: Steve John, Jack John (would not match Jack Johns).

The @ character works like .., but matches just a single character.

When performing a zoom, Paradox does distinguish between uppercase and lowercase letters. When using a pattern matching character, however, Paradox does not make this distinction.

## Adding Data To Tables

There are two ways to add data to a table. You can edit the table, and then use the Insert key to insert new blank rows for data, or add new rows at the bottom of the table. Or, you can use the *Modify Dataentry* command.

Generally, when adding more than one or two records, you should use the *Modify Dataentry* command. One reason is that when using editing to add records, you have access to all the other records of the table,

and you can accidentally change or delete those records. When performing data entry through *Modify Dataentry*, you enter data to an empty copy of the table.

To add records to a table using the *Modify Dataentry* command, follow these steps:

- 1) Issue the *Modify Dataentry* command.
- 2) Specify the table that you want to add data to and press the Enter key.
- 3) Enter records. You can press *F7 Form Toggle* to enter data using form view.
- 4) When finished, press *F2 Do\_It!* or select the *Do\_It!* command from the menu.

When adding data to a table that contains key fields, the data that you're entering may contain duplicate keys. Paradox places these records in a temporary table called *Keyviol*. You should edit these values and add them to the original table.

---

## Miscellaneous Commands

---

### Renaming

---

The *Tools Rename* command lets you rename Paradox objects. The most common use is *Tools Rename Table* to rename a table. For example, after using a query to develop an *Answer* table, you might want to rename it. Otherwise, since *Answer* is a temporary table, Paradox will delete it very soon.

### Copy

---

The *Tools Copy* command makes a copy of a table. Besides making a copy of the table, this command also copies the table's family of objects—reports, forms, validity checks, image settings, indexes, and so forth. You can copy the table in the same directory with a different name, or by specifying a different drive or directory name, copy the table and its family to a different directory.

The *Tools Copy JustFamily* command copies just the family of objects—reports, forms, validity checks, image settings, indexes, and so forth from one table to another. The source and target table must have similar structures. This command is useful when you perform a query on a table to produce a subset of the original table's data. To print one of the reports that belong to the original table on the *Answer* table's data, you'll have to copy the report from the original table to the *Answer* table. The *Tools Copy Report* command will copy a single report, but *Tools Copy JustFamily* will copy all reports.

### Delete

---

The *Tools Delete* command deletes tables, or reports or forms from a table. Deleting a table deletes the data and its family of objects.

## Info

---

The *Tools Info* command provides various information about your data. The most useful of these commands is *Tools Info Structure*, which lists the structure of a table. This command creates a table called *Struct*, which you can manipulate and print just like any other Paradox table.

## Empty

---

The *Tools More Empty* command empties a table of its data. It does not delete the table or any of its family members, it simply deletes all the data from the table.

## Add

---

The *Tools More Add* command adds records from one table to another table. The two tables must have a similar structure. Compatible means that the tables have the same type of fields in the same order. The fields don't have to have the same names, but must be the same field types.

When performing this command, specify the source and target tables. If the target table does not have key fields, Paradox adds the records from the source table to the end of the target table.

If the target table contains key fields, Paradox displays a menu with two choices. If you select the *NewEntries* command, Paradox adds the records from the source table to the target table. If a record in the source table has the same key value as a record already in the target table, Paradox places the record from the source table in the *Keyviol* table. You should then examine the records in the *Keyviol* table, edit them if necessary, and add them to the target table again.

If you select the *Update* command from the menu, Paradox performs these steps: For each record in the source table, Paradox finds its match in the target table, based on the values in the key fields. Paradox then replaces the matching record in the target table with the record in the source table. If no record in the target table matches a record in the source table, Paradox adds the record to the target table as a new record.

## Exercises

---

### Editing Data

Make these changes to the *People* table. Try to make the changes as effortlessly as possible. Use the zooming technique when possible to find the records that you need to change.

- 1) Marilyn Keegan has married and wishes to take her husband's name, which is Smith. Make the necessary changes and store them permanently.

Answer: View the *People* table and press *F9 Edit*. Move to the *Last Name* field and press *Control-Z Zoom*. Type Keegan and press the Enter key. Make sure you're at the right Keegan; if not, press *Alternate-Z Zoom Next* as needed. Press *Control-Backspace* to delete the entire field and type the new name Smith. Move to the *Honorific* field and make the necessary change. Finally, press *F2 Do\_It!* to record the change permanently.

- 2) It has come to your attention that the word *Mr.* was misspelled several times as *Mz.* Find these records and correct them. Don't make your changes permanent just yet.

Answer: View the *People* table and press *F9 Edit*. Move to the *Honorific* field and press *Control-Z Zoom*. Type *Mz.* and press the Enter key. Make the changes as needed, pressing *Alternate-F9 Zoom Next* to find additional records to fix.

- 3) The one person who resides in Michigan (abbreviated MI) has passed away. Remove this person from the table, but do not make the changes permanent yet.

Answer: Move to the *State* field and use *Zoom* to find this record. Press the Delete key to delete it.

- 4) You find that you made a mistake in deleting the previous records. Restore the record, then make the editing changes permanent.

Answer: Press *Control-U Undo* or select the *Undo* command from the menu. Press *F2 Do\_It!* to make the changes permanent.

- 5) Enter editing mode for the *People* table again. "Accidentally" lean on the Delete key for a few seconds. Recover all the deleted records.

Answer: Select the *Cancel* command from the menu. Select *Yes* to confirm.

### Creating Tables

- 1) Create a table to store information about parts carried in inventory. Information needed includes an eight-digit part number that contains alphabet and punctuation characters, a 12-letter department name, cost, price, and quantity on hand values, and the date a shipment of this part was last received. The part number should be unique, so make it a key field. Call the table *Inv1*.

Answer: Issue the *Create* command and supply *Inv1* as the table name. Fill in values like this:

STRUCT	Field Name	Field Type
1	Part Number	A8*
2	Department	A12
3	Cost	\$
4	Price	\$
5	Quantity On Hand	N
6	Last Received	D

Note the asterisk after the *Part Number* field type to indicate the key field. Press *F2 Do\_It!* when finished.

- 2) Add the following data to the table:

Part Number	Department	Cost	Price	On Hand	Date
A-1000	Toys	12.95	22.75	12	10/25/89
A-1500	Toys	3.25	6.95	53	10/26/89
A-1000	Food, Exotic	1.00	2.50	2	today's date
A-1200	Food	.45	.95	142	today's date

Answer: Issue the *Modify DataEntry* command and specify *Inv1* as the table to enter data to. Type the data. Make sure to use the *Control-D Ditto* key to duplicate values from the previous record. Press the spacebar three times to enter today's date. Press *F2 Do\_It!* when finished.

- 3) Because the table is keyed and we entered a duplicate part number value, Paradox created the *Keyviol* table to hold the duplicate record. Change its part number to *A-1100* and add it to the table.

Answer: Edit the *Keyviol* table and change the part number. Press *F2 Do\_It!* to complete the edit. Use the *Tools Mode Add* command, specifying *Keyviol* as the source table and *Inv1* as the target table. Select the *NewEntries* command to add this record as a new record to the *Inv1* table.

- 4) Restructure the *Inv1* table, making these changes: Move the *Last Received* field ahead of the *Cost* field. Make the *Department* field 10 characters long. Add a field at the end of the table for a description. This field should allow for 40 characters.

Answer: Issue the *Modify Restructure* command, specifying *Inv1* as the table to restructure. Move to the *Cost* field and press the Insert key to insert a blank row. Type the name of the *Last Received* field and press the Enter key to move the field. Edit the *A12* after the *Department* field to *A10*. Move to the end of the table and add the description field. Press *F2 Do\_It!* when finished. Select the *NoTrimming* command so that Paradox won't trim the *Department* field values. Then deal with the *Problems* table. You'll want to edit the offending value in some way, then use the *Tools More Add* command to add the records from the *Problems* table to the *Inv1* table.

### Image Settings

- 1) Adjust the widths of the numeric fields of the *Inv1* table to smaller values. Make the settings permanent.

Answer: View the table and use the *Image ColumnSize* command for each column that you want to change. Issue the *Image KeepSet* command to make the changes permanent.

2) Format the *Quantity On Hand* field as General with zero decimal places.

Answer: View the table and use the *Image Format* command. Move to the *Quantity On Hand* field and press the Enter key. Select *General* from the menu of formats. Backspace over the 2, type 0 and press the Enter key.

3) Move the *Description* field before the *Cost* field. Make the change permanent.

Answer: View the table and issue the *Image Move* command. Select *Description* as the field to move. Move to the *Cost* field and press the Enter key. Issue the *Image KeepSet* command to make the change permanent.

---

## Queries

---

One of the most important actions database software performs is to query the database. A query is often thought of as a question about the data in a table (or perhaps several tables), but Paradox queries go beyond that to let you perform many actions with your data.

The basic steps in making a query are:

- 1) Use the *Ask* command from the menu and specify the table you want to query. This action produces the query form.
- 2) Fill in the query form.
- 3) Process the query by pressing *F2 Do\_It!*.

Most queries produce an additional table. The most common query, asking a question about the data, produces a table called *Answer* that contains the selected data. Other queries, such as a *changeto* query, modify data in the table itself.

---

## Data Selection Queries

---

The most common type of query is the data selection query, which produces an *Answer* table that contains selected records from the source table.

The first step in making these queries, as with all queries, is to produce the query form by using the *Ask* command from the menu. Paradox asks for the name of the table to query, so type the name of the table or press the Enter key to select the table name from a menu of tables.

Once the query form appears, you must tell Paradox which fields or columns of data to include in the *Answer* table. You may select all fields, or just the fields of interest. Function key F6, called the *Checkmark* key, places a check to indicate that you want to include the field in the *Answer* table. The *Checkmark* key is a toggle, so if you've checked a field and then change your mind, move to that field and press *Checkmark* again.

A shortcut is to move to the very left column of the query form (beneath the name of the table that this query form is for) and press *F6 Checkmark*. This checks all fields.

Again, remember that only checked fields appear in the *Answer* table. A second important point is that the *Answer* table does not contain duplicate records. For example, if you had a table of all people in the United States and checked just the *State* field, only 50 records would appear in the *Answer* table, even though there are over 200 million records in the source table. That's because considering only the field that's checked, there are just 50 unique records, one for each state in the country. If you checked just the field that held the person's area code, you'd expect 200 records in the *Answer* table (assuming there are 200 area codes in the United States). If you checked all fields, you'd expect 240 million records in the *Answer* table, because considering all fields of the records, no two Americans are entirely unique (considering Social Security Number assures this).

If you do want to include duplicate records in the *Answer* table, use *Alt-F6 Check Plus* instead of *F6 Checkmark*. *Check Plus* includes duplicate records. If you used *Check Plus* to check the *State* field as

the only checked field in the query, then you would have 240 million records in the *Answer* table—one record for each record in the source table—even though there are many duplicates.

A third point is that Paradox sorts the records in the *Answer* table in ascending order if you've used *Checkmark* to check fields. If you use *Check Plus*, however, Paradox does not sort the *Answer* table.

To select certain records from the source table, fill in the query form with *conditions* that the data must satisfy. The first type of condition is the exact match, where you want to find records that match a certain value. For example, to find all the data about Johnny Carson from the *Buyers* table, fill out a query form like this:

BUYERS	Name	Address	City	State
	√ Johnny Carson	√	√	√

Note that all fields are checked, (this screen can't show all of the fields, however) since we wanted to include all the information about Johnny Carson. In the *Name* field of the form, we typed *Johnny Carson* since that's the data we wanted. This query creates an *Answer* table with exactly the same structure as the source table (*Buyers*), but containing just a single record.

If we had wanted to know just Johnny Carson's address, we might have created this query form:

BUYERS	Name	Address	City	State
	Johnny Carson	√		

Note that the only field checked is *Address*, so the *Answer* table contains just that field. It might be a good idea to go ahead and check the *Name* field, as in this query:

BUYERS	Name	Address	City	State
	√ Johnny Carson	√		

This time, the *Answer* table will contain the *Name* and *Address* fields, and we can check to make sure that it's Johnny Carson's address that we found. It is not necessary, however, to check fields where you place selection criteria.

Paradox also includes pattern operators that help match data. The most useful pattern operator is .. (two periods), which match any number of characters. Here's a few examples of the use of this pattern operator used in the *Name* field:

John..	Matches any name beginning with John: Johnny Carson, John Wilson.
..John..	Matches any name with John somewhere in it: Steve Johnson, Jack Johns, Johnny Carson.
..John	Matches any name ending with John: Steve John, Jack John (would not match Jack Johns).

The @ character works like .., but matches just a single character.

The special operator *like* is especially useful in queries. Normally, Paradox pays attention to capitalization during queries and treats a lowercase *a* different from the capital *A*. Thus, searching for *CA* when the data was entered in lowercase won't produce matches. But using the condition *like CA* works, as the *like* operator ignores capitalization. Also, *like* can find inexact matches based on spelling. For example, if you didn't know whether Johnny Carson spelled his name Carson or Carsen, using the condition *like Johnny Carson* would find both spellings.

The word *blank* finds fields which are blank. To use this, just type the word *blank* under the field name. The word *not* reverses the meaning of a query condition. For example, to find the records for everyone who doesn't live in Kansas, type *not KS* for the query condition. To find those records where a field value is not blank or empty, use *not blank*.

Suppose now we'd like to perform a slightly more complex query, finding the residents of Beverly Hills, California. We can't be assured of complete accuracy by simply writing *Beverly Hills* beneath the *City* field, as there are certainly cities in states besides California called Beverly Hills. In this example, the records must match two condition at the same time: the *City* must be Beverly Hills, and the *State* must be CA. This query accomplishes this:

BUYERS	Name	City	State	Zip
√		√ Beverly Hills	√ CA	

On the same row of the query, we write both conditions for the *City* and *State* fields. For Paradox to include a record in the *Answer* table, both conditions must be met at the same time.

For this query, the *Answer* table contains the *Name*, *City*, and *State* fields, since those three fields are checked.

In the next example, suppose you'd like to find all people in both California and New York. This example is different from the previous, where a record had to match two conditions at the same time. In this example, we'd like to include a record in the *Answer* table as long as it matches one of two conditions—*State* is CA or *State* is NY. This query form will do perform this:

BUYERS	Name	City	State	Zip
√			√ CA	
√			√ NY	

In this example, the query form includes two rows. The first row includes those records where the *State* is CA, and the second row includes records where the *State* is NY. As long as any record matches one of the rows in the query form, Paradox includes it in the *Answer* table.

We can see that query conditions on the same row of the query form are joined by the logical operator *and*, while conditions on separate rows of the query form are joined by *or*.

Note that the *Name* field is checked on both rows of the query form. The query form can have as many rows as you like, so you can create additional rows to include other states in the *Answer* table.

Paradox version 3.0 makes this type of *or* query easier to perform by introducing the keyword *or*, which works like this:

BUYERS	Name	City	State	Zip
√			√ CA or NY	

While this form of this type of query is more direct, it's sometimes necessary to use the first form.

## Range Queries

Suppose you'd like to find records with a certain range of values, perhaps finding those people with *Credit* over one million dollars. This query asks that question:

BUYERS	Name	Credit
√		>=1000000

The condition is  $>1000000$ , so records in the *Answer* table will match that condition. (Note that you can't use commas in the numbers entered in the query forms.) Paradox uses these range operators in queries:

=	Is equal to. This is not necessary, as if you typed 1000000 as the condition, Paradox would find records with a credit limit of 1000000, just as typing CA beneath State finds records with the state of CA.
> <	Greater than and less than. (Use the shift key with the comma and period for these symbols.)
>= <=	Greater than or equal to and less than or equal to.
not =	Not equal to a number.

Make sure you distinguish between the pairs  $>$  and  $>=$  and  $<$  and  $<=$ . Using  $>1000000$  finds those people with *Credit* of over one million, but not those with *Credit* of exactly one million. Using  $>=1000000$  includes those people with *Credit* of exactly one million.

You can combine range the range operators to make narrower selections. For example, suppose you want to find those people with *Credit* between one and two million dollars. This query asks that question:

BUYERS	Name	Credit
√		>=1000000, <2000000

This query uses two ranges separated by a comma, so records must match both conditions for inclusion in the *Answer* table. The first condition,  $>=1000000$  matches records of one million or greater, and the second condition  $<2000000$  matches records less than two million. Since the conditions are on the same row of the query form and separated by a comma, records must match both at the same time, and the query works.



ANSWER	Major Region	Sum of Population
1	Midwest	58867093
2	Northeast	49136741
3	South	74738592
4	West	43173422

The *Answer* table contains four rows, because there were four unique values in the checked field. The new field, *Sum of Population*, contains the sum of the *Population* field for all the records in each *Major Region*.

The summary operators that you can use in this way are *average*, *count*, *max*, *min*, and *sum*. Each can operate on unique or all values by following the operator with the words *unique* or *all*. When using *unique*, the summary operator summarizes unique records only. When using *all*, the operator uses all records, including duplicates. For example, consider these two queries with their corresponding *Answer* tables:

STATES	State
√ Calc Count Unique	

ANSWER	Major Region	Count of Major Region
1	Midwest	1
2	Northeast	1
3	South	1
4	West	1

STATES	State
√ Calc Count All	

ANSWER	Major Region	Count of Major Region
1	Midwest	12
2	Northeast	9
3	South	16
4	West	13

In the first example, the word *unique* means to count unique values only. Since for all records in the Midwest region have the same *Major Region* value, and we instructed Paradox to count unique values in the field only, the count is 1 for each region.

The second example tells Paradox to include all values, even duplicates, in the count. Therefore, Paradox counts the regions accurately.

## Examples in Queries

---

Paradox queries contain a powerful tool called *examples* that let you link together several table in a single query, or even link different rows of the same table. To start the study of examples, let's look at the *Students* and *Classes* tables.

Structure of the *Students* table:

---

STRUCT	Field Name	Field Type
1	<b>Sn</b>	<b>A4</b>
2	<b>Last Name</b>	<b>A15</b>
3	<b>First Name</b>	<b>A15</b>
4	<b>Address</b>	<b>A20</b>
5	<b>City</b>	<b>A14</b>
6	<b>State</b>	<b>A2</b>
7	<b>Zip</b>	<b>A5</b>
8	<b>Areacode</b>	<b>A3</b>
9	<b>Phone</b>	<b>A8</b>
10	<b>Enrolled</b>	<b>D</b>
11	<b>Total Hours</b>	<b>N</b>
12	<b>Total Grade Points</b>	<b>N</b>
13	<b>Gpa</b>	<b>N</b>

Structure of the *Classes* table:

STRUCT	Field Name	Field Type
1	<b>Sn</b>	<b>A4*</b>
2	<b>Department</b>	<b>A12*</b>
3	<b>Level</b>	<b>A3*</b>
4	<b>Term</b>	<b>A1*</b>
5	<b>Year</b>	<b>A2*</b>
6	<b>Grade</b>	<b>A1</b>

These tables were created with the *Tools Info Structure* command.

You can see that the *Students* table contains the biographical or personal information about each student—name, address, and so forth. The *Classes* table contains information about classes taken by all students. Each student is identified by a student number (the field *Sn*), and so too each class in the *Classes* table is associated with a student number.

Our question is to develop a list of all classes that William Smith has taken. No single table can provide the answer, as no table contains information about classes and also student names. But combining the *Students* and *Classes* tables together can provide the answer, as in this query:

STUDENTS	Sn	Last Name	First Name	Address
	<u>a</u>	Smith	William	

  

CLASSES	Sn	Department	Level	Term
	<u>a</u>	√	√	

There are two query forms in this query, one for the *Students* table and one for *Classes*, requiring two separate *Ask* commands. We can already understand two of the condition in the *Students* query form that select the record for William Smith. The third condition, the a beneath the *Sn* (student number) field, is an example. Example elements serve to link one table with another. Examples can be anything you want—the use of a is an arbitrary choice. To enter an example element, you first press *F5 Example* and then type the example. On the computer screen, example elements appear in reverse or highlighted video or in a different color, while in printed or stored queries, they're preceded by an underscore character. In the examples in this booklet, example elements appear in double underline.

The query form for the *Classes* table contains the same example element a beneath its *Sn* field, and checks for the *Department* and *Level* fields.

The example element a serves to link the two tables together. The example stands for whatever value is selected from the table. In this case, the example a stands for William Smith's student number. In effect, this query says *Find the student number for William Smith and store it as a. Then, move to the Classes table and find all classes where the associated student number is the value stored in a.*

The *Answer* table for this query will contain just two fields, the *Department* and *Level* fields from the *Classes* table, because those are the only fields checked. Fields used in an example do not necessarily need to be checked.

A second example query: Suppose you want to find the names of all students who have taken Chemistry 101. These query forms provide the answer:

STUDENTS	Sn	Last Name	First Name	Address
	<u>bob</u>	√	√	

  

CLASSES	Sn	Department	Level	Term
	<u>bob</u>	<b>Chemistry</b>	<b>101</b>	

In this query, the conditions in the *Classes* table find those records where the class is Chemistry 101. The example bob links the student numbers together, so that each time Paradox selects a record from the *Classes* table, it finds all records from the *Students* table with matching student numbers.

Examples also find use when querying single tables. For example, find those people with *Credit* equal to or greater than Johnny Carson's from the *Buyers* table. We could perform a query to find out Johnny Carson's *Credit*, and then another query using that value, but this single query performs the trick:

BUYERS	Name	Credit
	<b>Johnny Carson</b>	<u>x</u>
	√	>= <u>x</u>

The first line of the query finds the record for Johnny Carson and assigns the value from the *Credit* field the example name x. The second row returns all *Names* (because that field is checked) and *Credit* values (again, a checked field) for where the *Credit* is greater than or equal to the example x.

Another example: Who is older than Johnny Carson?

BUYERS	Name	Birthday
	<b>Johnny Carson</b>	<u>birthday</u>
	√	< <u>birthday</u>

The first line finds Johnny Carson's *Birthday* and associates it with the example birthday. The second line finds all records where the *Birthday* field is less than (meaning before in time) to the example birthday.

## Calculations in Queries

Examples let us perform calculations in Paradox queries. The special operator *calc*, followed by an expression, results in a new field in the *Answer* table, containing a calculated expression. The calculated expression almost always makes use of examples.

For example, suppose you'd like to see what the *Credit* field values would look like if you increased them by ten percent. This query performs the calculation:

```
BUYERS-----Name-----Credit-----
||√||                               ||√ a, calc a * 1.1||
|||
```

The *Answer* table contains three fields: the *Name* and *Credit* fields because of the checks, and a new field called *Credit \* 1.1* which contains each *Credit* multiplied by 1.1, as the formula indicated. The example element a serves as a label for the *Credit* for use later in the formula. No data in the source table (*Buyers*) changed.

## Changing Data in Tables

Suppose that after reviewing the results of the increased credit calculations you decide you'd like to change the actual data in the table to the new *Credit* values. A *changeto* query lets you do just that. For each field you'd like to change, use the word *changeto* followed by an expression or value. Paradox makes the change to the original table, and also creates a temporary table called *Changed*. This table contains an original copy of all the data that was changed, so that if you're not satisfied with the results, you have a backup.

This query will increase all *Credit* by ten percent:

```
BUYERS-----Credit-----
||√ a, changeto a * 1.10||
|||
```

Again, the example element a serves as a label for the *Credit* values so that we can refer to them in the formula after the *changeto* operator.

This query changes the *Credit* values for all the records. You could, of course, narrow the action of the query to just people in California by writing *CA* under the *State* field, for example. Or to increase the *Credit* just for those who already have *Credit* of five million or greater, use this query:

```
BUYERS-----Credit-----
||√ a, >=5000000, changeto a * 1.10||
|||
```

This query first assigns the example a to the *Credit* field, then selects just those *Credit* of five million or greater, then performs the change on just those records.

## Saving Queries

---

To save the query forms that you generate, create the query, then issue the *Scripts QuerySave* command. Supply a DOS file name to store the query in.

To replay the query, issue the *Scripts Play* command. Select the script name that you used and press the Enter key. Note that the query forms appear on the screen. To perform the query as is, press *F2 Do\_It!*. If you need to, you can make adjustments to the query, and then press *F2 Do\_It!* to produce the result. If you make corrections to the query form and want to save the new forms, issue the *Scripts QuerySave* command and reuse the same script file name. Issue the *Replace* command to overwrite the existing script file with the new one.

Remember that the *Answer* table that a query produces does not last for very long. The next query that generates an *Answer* table destroys the existing *Answer* table. Exiting from Paradox or using the *Tools More Directory* command to change the working directory also deletes the *Answer* table. If you'd like to preserve an *Answer* table permanently, use the *Tools Rename Table* command to rename the *Answer* table to some other name.

Renaming the *Answer* table is usually not productive, because as soon as you change data in the table that the *Answer* table derived its data from, the *Answer* table becomes obsolete. Saving the query forms is usually better. Then, the next time you need to see the results of the query, use the *Scripts Play* command to reconstruct the query forms, press the *F2 Do\_It!* key to reprocess the query, and view the up-to-date *Answer* table.

## Query Exercises

- 1) In the *People* table, produce all the data for just the people living in California.

Answer: Issue the *Ask* command for the *People* table. Under the table name, press *F6 Check* to check all fields. In the *State* field, type *CA*. Press *F2 Do\_It!* to create the *Answer* table.

```

PEOPLE-----First Name-----Last Name-----State-----Zip-----
      |√                |√                |√ CA                |√
      ||                ||                ||                ||
      ||                ||                ||                ||
  
```

- 2) In the *People* table, produce all the data for just the people living in San Francisco, California.

Answer: Issue the *Ask* command for the *People* table. Under the table name, press *F6 Check* to check all fields. In the *City* field, type *San Francisco*. In the *State* field, type *CA*. Press *F2 Do\_It!* to create the *Answer* table.

```

PEOPLE-----First Name-----Last Name-----City-----State-----
      |√                |√                |√ San Francisco    |√ CA
      ||                ||                ||                ||
      ||                ||                ||                ||
  
```

- 3) In the *People* table, produce just the names of the people living in California or Texas.

Answer: In the query form for the *People* table, check the *First Name* and *Last Name* fields. Move to the *State* field and type *CA or TX*. Press *F2 Do\_It!* to process the query. Is it necessary to check the *State* field? No it isn't, as the question didn't ask for the state field in the *Answer* table. But you might want to check the *State* field to provide a method for checking your answer. In real life, remember, the amount of data may be so large that you can't manually check the *Answer* table for correctness.

```

PEOPLE-----First Name-----Last Name-----State-----Zip-----
      |√                |√                | CA or TX          | | | | |
      ||                ||                ||                ||
      ||                ||                ||                ||
  
```

- 4) Produce a list of all the states that the people in the *People* table live in.

Answer: Check the *State* field only.

```

PEOPLE-----State-----
      |√                |
      ||                ||
      ||                ||
  
```

- 5) Count how many people live in each state.

Answer: Check the *State* field only, and type *Calc Count All* in the *State* field.

PEOPLE	State
√ Calc Count All	

- 6) Produce a list from the *People* table of all the cities that people live in. Count how many are in each city.

Answer: Check the *City* and *State* fields. (You need to check the *State* field so that cities like Bloomington, Illinois and Bloomington, Indiana are counted separately.) In the *City* field, type *Calc Count All*.

PEOPLE	City	State
√ Calc Count All	√	

- 7) In the *States* table, find those states where the largest city is also the capitol city.

Answer: Ask about the *States* table. Check the *State*, *Largest City*, and *Capitol* fields. In the *Capitol* field, press *F5 Example* and type *x*. In the *Largest City* field, press *F5 Example* and type *x*. Press *F2 Do\_It!*.

In this query, the example element *x* serves to represent a city name. Since the same example element is used in two different fields, the example elements are joined together with the logical operator *and*, and the two city names must be equal for the record to be in the *Answer* table.

STATES	State	Capitol	Largest City
√	√ x	√ x	

- 8) In the *States* table, the *Dukakis* and *Bush* fields contain the vote totals for those candidates in the 1988 presidential election. The *Reagan* and *Mondale* fields contain the totals in the 1984 election. Find those states that Reagan won in 1984, but Bush lost in 1988.

Answer: Create a query form like this:

STATES	State	Dukakis	Bush	Mondale	Reagan
√	<u>duke</u>	< <u>duke</u>	<u>fritz</u>	> <u>fritz</u>	

An entry like duke means an example element.

First, the example duke assigns that example to Dukakis's vote totals. In the *Bush* field, >duke finds the records where *Bush* is greater than *Dukakis*. These are the states that Bush won in 1988. Similarly, the example fritz in the *Mondale* field assigns that example meaning to Mondale's vote totals, and in the *Reagan* field, >fritz finds the states the Reagan won over Mondale. Since these are on the same row of the query form, records must meet all condition to be included in the *Answer* table.

- 9) From the *States* table, produce the presidential vote totals for Bush and Dukakis.

Answer: Type *Calc Sum All* in both the *Bush* and *Dukakis* fields. Don't check any fields.

STATES	Dukakis	Bush
	Calc Sum All	Calc Sum All

10) The *Purchase* table contains purchase history data for parts in an inventory. There are many instances of each part being purchased. Find, for each part, the number of purchases made, the date of the last purchase, the average price paid, the highest price paid, and the lowest price paid.

Answer: Ask about the *Purchase* table and fill in the query form like this:

PURCHASE	Part Number	Order Date
	√	Calc Count All, Calc Max All

  

Price
Calc Average All, Calc Min All, Calc Max All

The check in the *Part Number* field produces one row in the *Answer* table for each part that was purchased. The *Calc Count All* in the *Order Date* field counts the dates that orders were placed on, producing the number of orders placed for each part. Since dates in Paradox become larger as time goes on, *Calc Max All* finds the largest *Order Date* for each part, producing the date of the last order. In the *Price* field, the calculation operators calculate the average, smallest, and largest price for each part.

11) From the *Students* and *Classes* tables, find the names of the students who took Chemistry 101.

Answer: Create query forms like this:

STUDENTS	Sn	Last Name	First Name	Address
	<u>s</u>	√	√	

  

CLASSES	Sn	Department	Level	Term
	<u>s</u>	Chemistry	101	

In the *Classes* table, the criteria in the *Department* and *Level* fields find the students who have taken Chemistry 101. The example element s in the *Sn* field in both tables links the two tables together on common student numbers. The checks in the *Students* table provide the fields to include in the *Answer* table.

12) The *Vendors* table contains vendor numbers and names of the vendors referenced in the *Purchase* table. List the names of the vendors who have supplied part number A-3300.

Answer: Ask about the *Purchase* and *Vendors* tables and fill in the query forms like this:

Paradox Level 1

---

PURCHASE	Part Number	Vendor	Order Date	Arrival Dat
	A-3300	<u>v</u>		

VENDORS	Vendor	Company	Contact	Address
	<u>v</u>	✓		

In the *Purchase* table, fill in the part number we're looking for. The example element v provides a link to the *Vendors* table. In the *Vendors* table, the example element v establishes the link to the *Purchase* table. The check in the *Company* field places that field in the *Answer* table. Since the question asked only for the names of the companies, this is the only field we need to check. If we wanted address and telephone numbers, we could check additional fields in this table.

## Reports

---

Reports are another important skill to develop when using Paradox. A report is a design or schematic for printing data. Some of the features that report designs can use include flexible layout, placement of any fields in any order, sorting of data, grouping data, performing calculations, performing summaries, referencing data in other tables, and many other tasks.

Reports are associated with a particular table, and are known as part of the table's family of objects. You can use a report design with similar tables by copying the report to the additional table. Each table may have up to 15 reports, known as report *R* and reports 1 through 14. Report *R* is a special report known as the instant report.

To design a report, follow these steps:

- 1) Issue the *Report Design* command.
- 2) Select the table that the report will belong to.
- 3) Select the number for this report. Generally, let the first report be number one, with the second report number two, and so forth.
- 4) Supply a description for this report. The description is up to 40 characters of text that describes the report. The description is optional, but a good idea, especially when a table has several reports.
- 5) Identify whether the table is tabular or free-form in design. Tabular reports contain data in columns. Free-form reports don't have columns. Most of the reports you create will likely be tabular reports. A mailing label is an example of a free-form report.

At this point, you're viewing the Paradox report design workspace. Your job is to place different report objects on the design workspace to produce the report that you want. When you're finished designing the report, press *F2 Do\_It!*. This action saves the report design and returns to main Paradox mode. You can then print the report.

### The Paradox Standard Report

---

Here's an example of what a tabular Paradox report design might look like. The words in italics are not part of the report design, but instead are labels to identify parts of the report.

---

Designing report R1 for People table Report 1/4  
 Report Header  
 .....10.....20.....30.....40.....50.....60.....70.....8\*  
**(Report Header Band)**  
 page\_\_\_\_\_

**(Page Header Band)**

mm/dd/yy Credit Limit Report Page 999

---

table\_\_\_\_\_

**(Table Band)**

First Name	Last Name	MI	Suffix	Honorific	Saluta
AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAA	AAAAAAAAAA	AAAAAAAAAAAAAAAA	AAAAAA

---

table\_\_\_\_\_

**(Page Footer Band)**

page\_\_\_\_\_

**(Report Footer Band)**

---

Starting from the top of the screen, Paradox tells us we're designing report number one for the *People* table. The *1/4* tells us that we're looking at width one of four width total. These widths refer to page widths, as defined by the page width setting. Normally, this is 80 characters, so to print this report in full requires four page widths.

The line of numbers and dots is a ruler that provides a reference as to your position. The asterisk after the 8 (for position 80) defines the width of the page.

## Report Bands

---

The horizontal lines though the report divide the report into different bands. Each band is printed at a different time, and provides control over which report elements print at which times. As you move through the report design, Paradox displays the name of the band the cursor is in at the top left of the screen. The various bands with their descriptions follow:

Band	Meaning
Report Header	Paradox prints this band just once at the beginning of the report. Use the Report Header for the title of the report or a title page.
Page Header	Prints at the top of each page, and printed as many times as there are pages. Use the page header to print titles for each page, and also to control the top margin. The contents of this page header is the words Credit Limit Report (from the report description given) along with special field masks that print the date and page number.
Table Band	Paradox prints the table band for each record in the table. The contents of this table band are a blank line, and then two lines of field titles. The last line of the table band contains field masks that represent instruction to print data from the table at these locations.

Band	Meaning
	Actually, Paradox doesn't print the entire table band for each record in the table. Only the lines that contain field masks are printed for each record. This column band is divided into columns, one column for each field. This report, then, is a tabular report, as free-form reports don't have columns.
Page Footer	Printed at the bottom of each page. Use the page footer for footers at the bottom of each page, and to control the bottom margin.
Report Footer	Printed at the end of the report, but before the page footer of the last page. If the report is 50 pages long, Paradox waits until page 50 to print the report footer. Use the report footer for summary statistics about the data in the report, and for messages identifying the end of the report.

## Literal Text and Field Masks

Paradox reports can contain literal text and field masks. Literal text, like the words *Credit Limit Report* in the page header and the field titles and lines of dashes in the table band, print on the report at their position in the report design. Field masks, on the other hand, represent places where data from the table or other special data should print.

Field masks come in several types, as illustrated below:

A string like *AAAAAA* represents where a alphanumeric field will print. The number of *A*'s defines the width allocated to the field. If the field mask is not wide enough, part of the field may not print.

A string like *999999* represents where a numeric or currency field will print. Again, the number of *9*'s determines the space allocated for the field. If the field contains too many digits for the allocated space, Paradox prints asterisks in place of the number. A numeric field masks may contain commas and parenthesis if the field is formatted to contain them. The characters also indicate the number of decimal places that will display, as is *99999.99*, and also indicates whether parentheses and commas will be used, as in *(999,999.99)*.

A field mask like *mm/dd/yy* represents space for a date field. Date field masks may appear in other formats for date fields formatted with the different date formats.

In addition to field masks for fields from the table, Paradox lets you place field masks for special data items like the current date and time, and the page number.

Place field masks on the report design using the *Field Place* command. You can't type the field mask characters from the keyboard. As you move the cursor through a field mask, Paradox displays the field name at the top right of the screen.

The report design illustrated above is a Paradox standard report. Each time you design a new report, Paradox starts with this standard report. You can modify this as needed.

When designing reports, use these keys:

Insert	Toggles between insert and replace typing modes. Indicated at the top
--------	---

	right of the screen.
Control-Y	Deletes from the cursor position to the end of the line. To delete an entire line, place the cursor at the start of the line and press Control-Y.
Enter	When in insert mode, adds blank lines to the report.
Home, End	Moves to the first and last lines in the report.
Control-Home Control-End	Moves to the beginning and end of a line in the report design.
Control-Left Arrow Control-Right Arrow	Moves left and right a screen.
Tab Shift-Tab	When in the table band, moves from column to column.

## Example 1

---

As an example of report design, create a report that lists the first and last names of each person, along with their state and credit limit. The finished report design looks like this:

```
Designing report R1 for People table                      Report      1/4
Report Footer
...+...10...+...20...+...30...+...40...+...50...+...60...+...70...+...8*
```

—page—

### Credit Limit Report

Printed on mm/dd/yy at hh:mm pm

First Name	Last Name	State	Credit Limit
AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AA	(999,999,999.99)

—page—

---

The first step in designing this report is to issue the *Report Design* command. The table is *People*. Use report number one, let the description be *Credit Limit Report*, and make the report a tabular design.

The next step is to remove the unneeded fields. The command to use is *TableBand Erase* from the report

---

menu (accessed through the *F10 Menu* key). To use this command, select it from the menu. Then, in the table band, move to the column to remove and press the Enter key. Perform this command for all fields except those shown in the report design above.

Next, we want to remove the field mask for the current date in the page header so that we can place it at a different position. To remove a field, issue the *Field Erase* command. Move to the field mask that you want to remove and press the Enter key.

To remove the page numbering, move to the word *Page* and use the Delete key to remove the text. Use the *Field Erase* command to remove the page number field mask.

To place the *Printed on..* line, follow these steps: First, move to a line near the end of the page header and type *Printed on* and press the space bar. Then, to place the current date field, issue the *Field Place* command. Select the *Date* command from the menu of fields. Choose one of the date formats and press the Enter key. Then, use the cursor keys to move to where the beginning of the field mask should be. Then press the Enter key to place the field mask there. Move over a space and type the word *on*. Repeat the *Field Place* command, this time placing the time field using one of the two time formats.

To complete the report design, press *F2 Do\_It!*.

---

## Printing a Report

---

To print this report, issue the *Report Output* command. Select the *People* table as the table to print from. From the list of reports that appears, select report number one. (Notice that as you move to the *I*, Paradox displays the report description.) Select where to send the report—to the printer, the screen or a disk file.

When printing a report to the screen, you can press the Control-Break way to stop the output. You'll have to press any key two times, however, after pressing Control-Break. To stop a report that was sent to the printer, press any key. Paradox responds with a request to press *C* to stop the report, or *R* to resume printing.

---

## Printing a Subset of Data

---

Many times when you print a report, you want the report printed for a subset of the table's data, such as just for the people in Kansas. The Paradox reports, however, don't have a mechanism to restrict the records that are printed. Each time you print a report, Paradox prints all the data.

To print a report for just part of a table's data, perform a query on the table that creates an *Answer* containing just the data that you want to print. Then, print a report based on the *Answer* table.

This process, unfortunately, isn't quite as simple as it seems. Paradox reports, remember, are associated with a particular table. The *Credit Limit* report we created belongs to the *People* table, not the *Answer* table. Therefore, if you try to print a report from the *Answer* table, you'll get just the Paradox standard report, not the *Credit Limit* report.

The key is to copy the report from the *People* table to the *Answer* table. You can choose to copy just a single report, or to copy all the reports from the *People* table. Then, you can print the *Credit Limit* report based on the data in the *Answer* table. To illustrate this process, follow these steps:

- 1) Use the *Ask* command to get a query form for the *People* table. Check all fields in the form. (The shortcut is to press *F6 Check* when under the table name.) Under the *State* field, type *KS* to get the people in Kansas. Press *F2 Do\_It!* to process the query and generate the *Answer* table.
- 2) Issue the *Tools Copy JustFamily* command. Specify *People* as the source table and *Answer* as the target.
- 3) Select the *Replace* command to let Paradox replace the existing reports for the *Answer* table with the reports from the *People* table. Since *Answer* didn't have any reports, this doesn't actually erase any reports.
- 4) Issue the *Report Output* command. Specify *Answer* as the table. Select report number one, and then select *Screen* or *Printer* as appropriate.

If you reperform the query to produce a different subset of the *People* table, Paradox deletes the existing *Answer* table, and its family of reports. Therefore, to print the *Credit Limit* report from the new *Answer* table, you'll have to repeat the *Tools Copy JustFamily* command again.

You must check all fields from the *People* table when performing the query, even though the *Credit Limit* report makes use of just a few of the fields. This is because the *Tools Copy JustFamily* command requires that the two tables have the same structure.

An alternative command to use to copy the report is the *Tools Copy Report DifferentTable* command, specifying *People* as the source table and *Answer* as the target. This command copies just a single report, so the command asks which report to copy. Most of the time, it's easier to use *Tools Copy JustFamily* to copy all of the report from one table to another.

## Placing Totals

---

This report, which is similar to the first report, sums the *Credit Limit* field for the entire report. It also adds the page number in the page footer band.

Changing report R2 for People table

Report Ins 1/4

Page Header

.....10.....20.....30.....40.....50.....60.....70.....8\*

—page—

Credit Limit Report

Printed on mm/dd/yy at hh:mm pm

First Name	Last Name	State	Credit Limit
AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	AA	(999,999,999.99)

Page 999

—page—

Total Credit (999,999,999.99)

Since this report is similar to report number one for the *People* table, we can start by making a copy of that report. Call the new report number two, following these steps:

- 1) Issue the *Tools Copy Report* command.
- 2) Select the *SameTable* command, because we're copying the report to the same table.
- 3) Select report 1 as the source report, and report 2 as the target report.

Now, issue the *Report Change* command to let us make changes to the new report number two. This recalls the report design, letting us make changes to this report.

First, add the page number field to the page footer band. Move to the proper position and type *Page* and press the space bar. Use the *Field Place Page#* command to place the page number. Use the cursor keys to place the start of the field and press the Enter key. For this field, you can adjust the number of field mask digits used to reserve space for the page number. The three digits that already show should be sufficient, so press the Enter key.

Now, get ready to place the total of the *Credit Limit* field. First, consider when we want this total to print. It should print at the end of the report, after all the records on the table have been printed. The report footer band is the last band Paradox prints, so this is where we should place the total. The End key moves to the end of the report. At the left margin of the report, type the words *Total Credit*.

Now, we're ready to place the total of the *Credit Limit* field. This is a summary field—a field that performs a summary operation (sum, average, count, minimum, or maximum) of a field. Follow these steps to place a summary field:

- 1) Issue the *Field Place Summary* command.

- 2) Select *Regular*. A regular field is one that exists in the table, like the *Credit Limit* field we're working with. A calculated field is one that is calculated through a formula in the report.
- 3) From the list of fields, select *Credit Limit*. Pressing *C* narrows the list of fields to those beginning with *C*. Use the arrow keys to highlight the field and press the Enter key to select it.
- 4) Select the summary operation for this field. We want to add the values, so select *Sum*.
- 5) Select *Overall* to instruct Paradox to calculate this summary based on all the records printed so far. *PerGroup* is used when dividing the report into groups.
- 6) Move the cursor to where the field should begin. You'll want to align this field directly below the existing *Credit Limit* field mask in the table band. Place the cursor where you want the beginning of the field to appear. Press the Enter key after placing the cursor.
- 7) Adjust the number of digits to the left of the decimal place to appear and press the Enter key.
- 8) Adjust the number of digits to the right of the decimal place to appear and press the Enter key.

To add the line of dashes above the summary field, first create a blank line above the summary line. The easiest way to do this is to move to the beginning of the line containing the summary. (Control-Home does this quickly.) Then make sure that insert mode is on. (If necessary, press the Insert key.) Press the Enter key to insert a blank line. Then move to where the line should start and type the dashes as needed.

## **Adding Groups to Reports**

---

Grouping in Paradox lets you sort your data records into groups, and to print introductions and summaries for each group. The group feature is a very powerful feature of Paradox reports. This example groups the data in the *People* table by the *State* field. It prints the total of the *Credit Limit* field for each state, and the total of all the credit limits at the end of the report.

Changing report R3 for People table Report Ins 1/4  
 Report Header  
 .....10.....20.....30.....40.....50.....60.....70.....8\*  
 -----  
 -page-----

Credit Limit Report

Printed on mm/dd/yy at hh:mm pm

-----group State-----

Customers in AA: **(Group Header for State Field)**

First Name	Last Name	Credit Limit
AAAAAAAAAAAAAAAA	AAAAAAAAAAAAAAAA	(999,999,999.99)
Totals for AA		(999,999,999.99)

**(Group Footer for State Field)**

-----group State-----

Page 999

-----page-----

Total Credit	(999,999,999.99)
--------------	------------------

=====

In this report, you see two new bands—the group header for the *State* field, and the group footer for the *State* field. When Paradox prints the report, it follows this process:

- 1) Paradox prints the group header for the first state.
- 2) Paradox prints the table band for all the records belonging to the first state.
- 3) Paradox prints the group footer for the first state.
- 4) Repeat this process for all other states.

Since the group header prints before the data for a group, use it to introduce the group. Use the group footer, which prints after the data for a group, to summarize the group.

This report is similar to report number two for the *People* table. Use the *Tools Copy Report SameTable* command to copy report two to report three. Then, use the *Report Change* command to start modifying report number three.

The first step is to add the group for the *State* field.

This report contain a group based on the *State* field. Paradox prints the Group Header before each group, then prints the Table Band for each record in the group, then prints the Group Footer. To insert the group for the *State* field, follow these steps:

- 1) Issue the *Group Insert* command.
- 2) Our group is based on the contents of a field, so select *Field*.
- 3) From the list of field names, select *State*.
- 4) Use the cursor keys to position to cursor at the position for the group. In this example, place the cursor just above the top of the table band. Press the Enter key when there.

At this point, Paradox inserts the group header and group footer bands.

We'd like to introduce each state with a line like Customers in CA: To do this, follow these steps:

- 1) Move to the group header for the *State* field.
- 2) At the left margin, type *Customers in* and press the space bar.
- 3) Issue the *Field Place Regular* command. Select *State* as the field to place.
- 4) Position the cursor to where the field should start and press the Enter key.
- 5) Press the Enter key again to accept the proposed width of the field.
- 5) Type a colon after the field mask.

Now, when Paradox prints the group header for a state, it will introduce it. Because of the introduction, it's no longer necessary to include the *State* column in the table band. Use the *TableBand Erase* command to remove it.

After removing the column, note that the *Credit Limit* column moves to the left. The summary field for *Credit Limit* in the report footer band, however, didn't move, so it's out of alignment with the field it summarizes. To correct this, move to the report footer, and by using the Delete key, remove some of the spaces before the summary field and line of dashes. There is no automatic way of achieving perfect alignment.

## **Summarizing a Group**

---

To summarize a group, place summary fields in the group footer for a field. In this case, we'll place a summary field, along with some text, in the group footer for the *State* field. Follow these steps:

- 1) Move to the left margin of the group footer for state and type *Totals for* and press the space bar.
- 2) Use the *Field Place Regular* command to place the *State* field.
- 3) To place the summary field for the *State* field, issue the *Field Place Summary* command.
- 4) The *State* field is a regular field, so select the *Regular* command.
- 5) Select *Sum* as the summarization operator.

- 6) Select *PerGroup*. This makes Paradox reset the field for each state, so that each state's total credit limit is calculated separately.
- 7) Position the cursor at the beginning of the field and press the Enter key. Adjust the number of digits that you want to show.

To add the line of dashes above the field, insert a blank line before the field and type the dashes. It's also good to include a blank line as the last line of this group footer.

## Using a Calculated Field

Suppose that we require that we keep a credit reserve equal to 10% of the credit limits that we've granted. This reserve does not appear as a field in the *People* table. We can calculate this field, however, and print it on a report. The field never exists, but is calculated and printed only when printing this report. The design would look like this:

```

Changing report R4 for People table                                Report    1/4
Report Header
...+...10...+...20...+...30...+...40...+...50...+...60...+...70...+...8*

-----page-----

                                Credit Limit Report

        Printed on mm/dd/yy at hh:mm pm

-----group State-----
        Customers in AA:
-----table-----
        First Name      Last Name      Credit Limit      Reserve
        -----
        AAAAAAAAAAAAAA  AAAAAAAAAAAAAA  (999,999,999.99)  (999,999,999.99)
-----table-----

        Totals for AA                                (999,999,999.99)  (999,999,999.99)

-----group State-----

        Page 999

-----page-----

        Total Credit                                (999,999,999.99)  (999,999,999.99)
    
```

Calculated fields are made from expressions. An expression in a report may contain any field name, the arithmetic operators +, -, \*, and /, and constant values. Enclose field names in square brackets. The expression for the Reserve column is *[Credit Limit]\*.1*.

The first step in creating this report is to add a new column for the table band to hold the reserve field. To

do this, move to the table band after the *Credit Limit* field. Issue the *TableBand Insert* command and press the Enter key to place the new column.

Now, place the summary field following these steps:

- 1) Issue the *Field Place Calculated* command.
- 2) Type the expression for the calculation. When typing these expressions, enclose Paradox field names in brackets. The expression we need is  $[Credit\ Limit]*.1$ . Type it and press the Enter key.
- 3) Move to where the field should start. Start the field at the very beginning of the new column of the table band. Press the Enter key to place the field. Adjust the number of digits to show.
- 4) Move to the lines above the new field mask and type a column title and line of dashes.

If you need to make a column wider or narrower, use the *TableBand Resize* command. When Paradox asks you to move to the column to resize, move to the very rightmost position in the column. Then press the Enter key and use the arrow keys to make the column wider or narrower.

## Summarizing Calculated Fields

---

Paradox can summarize calculated fields as well as regular fields. To summarize the reserve calculation for each state, follow these steps:

- 1) Move to the group footer for state. Move under the reserve calculated field for proper alignment.
- 2) Issue the *Field Place Summary* command.
- 3) The field we want to summarize is a calculated field, so select the *Calculated* command.
- 4) Type the expression to summarize and press the Enter key. In this case, it's the same expression used before:  $[Credit\ Limit]*.1$ .
- 5) Select *Sum* as the summarization operator.
- 6) Select the *PerGroup* command to calculate this in a state-by-state basis.
- 7) Place the field and adjust the number of decimal places. Add a line of dashes before it.

To place this summary in the report footer, so that we calculate the total reserve needed for the entire report, follow the same steps. Instead of selecting *PerGroup*, select *Overall* to calculate the summary for the entire report.

## Commands for Reports

Command	Meaning
TableBand Resize	Lets you make a column wider or narrower. When Paradox asks you to move to the column to change, move to the very right edge of the column and press the Enter key. Then use the arrow keys to make the column wider or narrower.
TableBand Insert	Inserts a new column in the report. The new column appears to the left of the column you place the cursor in.
Field Place Regular	Places a regular field in the report. Regular fields are fields that exist in the table.
Field Place Calculated	Places a calculated field in the report. Calculated fields do not appear in the table, but are calculated from field values and constant values.
Field Place Summary	Places a summary fields in the report. Summary fields calculate statistics about a group in the report.
Field Place Date	Places the current date field in the report. Select the format you wish to use.
Field Place Time	Places the current time field in the report. Select the format you with to use.
Field Place Page	Places the page number field in the report.
Field Place #Record	Places the record number field in the report.
Group Insert	Inserts a new group. Place the cursor where you want to group inserted.
Group Insert Field	Creates new groups based on the contents of a field. Paradox prints the group header and footer once for each unique value in the field. If you create a group on the State field and the table contains data for 20 states, you'll get 20 groups.
Group Insert Range	Creates groups based on a range of data. For a numeric field, Paradox groups on a range of values that you specify. For a date field, Paradox can group on the da, week, month, or year. For a alphanumeric field, Paradox can group on the first few characters of the field values.
Group Delete	Deletes a group. Paradox deletes the group header and footer and their contents with this command, so be careful.
Group SortDirection	Specifies the sort order (ascending or descending) for the group.
Group Regroup	Lets you specify a different field for the group.
Field Place Summary	Places a summary field in a report. Most often, you place summary fields in Group Footers or the Report Footer to calculate summary statistics.
Field Place Summary Regular Field Place Summary	Places a summary of a regular or calculated field. Regular fields are those that appear in the table. Calculated fields are calculated with an expression.

Command	Meaning
Calculated	
Sum Average Count High Low	Specifies the statistic to calculate. Not all choices are available for all field types. For date fields, for example, you can't calculate the sum or average.
PerGroup	Calculates the summary statistic for just the records since the field was last calculated. PerGroup summary fields are usually placed in group footers to calculate statistics for just the records in the group just printed.
Overall	Calculates the summary statistic for all the data since the beginning of the report. Usually placed in the Report Footer to provide a grand total.
Field Reformat	Lets you make adjustments to a field mask's width. For numeric and date fields, lets you change the format style.
Field Justify	Lets you place a field left, right, or center within the width of the field mask.
Field CalcEdit	Enters an editing mode so that you can make adjustments to the expression of a calculated field.
Field WordWrap	Lets you place a long field on more than one line of a report.
Settings Margin	Sets the left margin, in characters, for the report.
Settings PageLayout Length	Adjust the length of the printed page. For most laser printers printing six lines per inch, set this to 60.
Settings PageLayout Width	Adjust the page width in characters.
Settings Setup	Selects setup strings to use for this report. You can use Paradox's predefined setup strings, or enter your own. Also selects the printer port to use.
Output Screen Output Printer Output File	Prints the report from within the report design. Output Screen is especially useful for previewing the report as you create it.

---

## Database Design

---

Database applications are some of the most important areas in which the computer can make an impact on a business. With the right database, a company can have nearly instant access to important information, and can analyze and present it in useful ways.

Database technology, however, is one of the most misunderstood areas of computer application. This article will introduce you to database systems, their applications, and guidelines for selecting and using database software.

## The Nature of a Database

---

A database system is a record-keeping system. Its job is to record information, maintain it, and present the information in a useful form. A database system consists of four main components:

- \* The computer hardware, which of course is the device that stores and manipulates the data. The computer may range in size from the smallest home computer to the largest mainframe computers.
- \* The database management software, often called the database software or database program. This is computer software that is responsible for all operations with the data. It may range from an inexpensive file management program like pfs:File to more expensive, full-featured systems like dBASE III Plus or Paradox. The database software is more important than the computer it's running on, as there is good and bad database software for most computers.
- \* The data itself. The data, of course, represents the whole reason for having a computer and database software.
- \* Users who will use the data. These people are the most important component of a database system, as the computer, database software, and data exist solely to serve their needs.

## Database Design

---

The design of your database tables is extremely important. With the correct design, it's usually easy to achieve the results you want. With an incorrect or inefficient design, it can be difficult or impossible to get the results you need.

## Classes and Students: One Bad Example

---

To illustrate the idea of a normalized database, consider a college that needs to track student enrollment. The important characteristic of this task is that each student enrolls in several classes each semester, and by the time each student earns a degree, they'll have enrolled in dozens of classes.

One database design that we might create looks like this (call this database A):

SN, Name, Address, Class1, Year1, Grade1, Class2, Year2, Grade2, Class3, Year3, Grade3,...Class50,

---

This database first keeps track of the Student Number (Sn), the student's name, and the address. In real life, we'd also keep track of the city, state, zip, telephone, birth date, parent's name, high school attended, and so forth. Then the database keeps track of class enrollment: Class1 to hold the title of the first class they take, Year1 to hold the year of enrollment, and Grade1 to hold the grade (left blank for now). Class2, Year2, and Grade2 hold information for the second class, and so forth on up to information for the 50th class.

This database, while intuitive, contains many problems and in practice is virtually worthless. Why?

- 1) What if the student enrolls in more than 50 classes? Where do we store the information for the 51st class? The only thing to do is create a second record for the student—creating a situation where most students have a single record in the database, but a few have two.
- 2) Many database programs have a limit on how many fields any record can have. In dBASE III Plus, the limit is 128 fields, and this database uses many more fields than that.
- 3) What about the student who enrolls in three classes and then drops out? This database stores many empty fields—using memory and disk space—for fields that will never be used.
- 4) What if we want to print a list of classes a student has taken, sorted by the department? We'll have to shuffle around the data in the record—an awkward process with most database programs.
- 5) This database is very difficult to query. Here's a sample question: Has student 1005 ever taken Chemistry 101? Since we have no way of knowing when any student takes any class (that is, Chemistry 101 could appear in class1, class2, on up to the class50 field), we'll have to formulate a query like this:

Display all records where SN = 1005 and (class1 = "Chemistry 101" or class2 = "Chemistry 101" or class3 = "Chemistry 101")...

Using the Query-By-Example technology in Paradox, the query looks something like this:

A	Sn	Class1	Class2	Class2
	1005	Chemistry 101		
	1005		Chemistry 101	
	1005			Chemistry 101
	1005			

Of course, you'll have to extend either of these queries on up to class50 = "Chemistry 101" in order to be complete. It goes without saying that this is unworkable, and it gets worse. Suppose now you need to accumulate an honor list of beginning Chemistry students—those students who received an "A" in Chemistry 101 in 1986:

Display all records where  
 (class1 = "Chemistry 101" and year1 = 1986 and grade1 = "A") or  
 (class2 = "Chemistry 101" and year2 = 1986 and grade2 = "A") or  
 (class3 = "Chemistry 101" and year3 = 1986 and grade3 = "A")...

The Paradox version, for just two classes out of the 50 total classes it is necessary to search in, looks like this:

A	Sn	Class1	Year1	Grade1	Class2	Year2	Grade2
		Chemistry 101	1986	A			
					Chemistry 101	1986	A

Again, we'd need to extend this query to include class50, year50, and grade50. Other situations that will be difficult to deal with include:

Print a class roster for each class.

Count how many classes each student has taken.

How many students enrolled in each class this year?

Print a report card or transcript.

Calculate grade point averages.

Each of these questions is nearly impossible to answer from this database, yet these are precisely the questions that the college was hoping to answer through this database.

## Classes and Students: Another Bad Example

---

Now, to illustrate a normalized database, consider this solution to the college enrollment task (call this database B):

SN, Name, Address, Class, Year, Grade

Remember, besides the Name and Address fields, we'd have other biographical fields like birth date, city, state, zip, telephone, and so forth.

Notice that each record in this database design has space for a single class enrollment. When a student enrolls in five classes in the new term, we'll add five of these records to the database. This means we'll have to type the name, address, city, state, zip, telephone, etc. five times too. Let's discuss the merits of this database, starting with the demerits first:

- 1) This database duplicates the biographical information many needless times. If a student eventually takes 50 classes, you'll carry 50 addresses and so forth in the database. This, of course, takes a lot of time to type and occupies a lot of storage space.
- 2) Eventually, the student will change addresses, and so records entered after that time will show the new address. Later, when it's time to print the student telephone book, we won't know which of the several addresses to use. This leads to the case of an inconsistent database. Database A doesn't have this problem, as all the biographical information is stored just once.
- 3) Suppose a new student enrolls in a single class and then cancels the class after the first week. If we delete the class enrollment from the database, we delete all the biographical information (the only instance of it) too.
- 4) We can't enter information about a prospective students until they actually enroll in classes. We could add a record to the database and leave the class, year, and grade fields blank, but a record with

a blank class title wouldn't make sense.

Now, for the good things about this database design.

- 1) There is no artificial limit on the number of classes a student can take, as we can keep adding records as needed.
- 2) We create only as many records as each student needs. If a student takes three classes and drops out, that student has just three records.
- 3) With this database, it's easy to print a sorted list of classes a student has taken.
- 4) This database is easy to query. The first question posed of database A (has student 1005 ever taken Chemistry 101?) is easy to answer:

Display all records where SN = 1005 and class = "Chemistry 101"

Either the record shows up or it doesn't. The Paradox version looks like this:

B	Sn	Class
	1005	Chemistry 101

The second question (accumulate an honor list of beginning Chemistry students—those students who received an "A" in Chemistry 101 in 1986) is easy, too:

Display all records where class = "Chemistry 101" and year = 1986 and grade = "A"

The Paradox version looks like this:

B	Sn	Class	Year	Grade
		Chemistry 101	1986	A

Other questions are easy to answer. For example:

Print a class roster for each class offered this year:  
Display all records where year = 1986 sorted by class.

How many students enrolled in a class this year?  
Count unique sn for year = 1986.

## Classes and Students: A Good Example

---

To model the enrollment information effectively we need to use two tables—one for the student information, and one for the class information. Here's the two database structures:

Database C:  
SN, Name, Address

Database D:

---

---

SN, Class, Year, Grade

These database designs store the two different types of information in databases designed just for that information: Database C stores just biographical information about students, and database D stores just enrollment information for classes. Good features of these designs include:

- 1) Each database contains just the right number of records to hold the data you need to track.
- 2) Each student has a single record in the C database. There is no needless duplication of information.
- 3) If a student's address changes, find the student's record in the C database and change it. Possibilities for database inconsistencies are reduced.
- 4) Each time a student enrolls in a class, the student gets a new record in the D database. If the student enrolls in four classes this semester, the student gets four new records in the D database. If a student eventually enrolls in 100 classes, that student has 100 records in the D database. If a student drops out after a semester, the student has just a few records in the D database and never has any more records added.
- 5) Because of this specialization of information in each database, each database is easy to query.

There are times, however, when we need to query both databases at the same time. This is where a linking field is necessary to provide a basis for matching records from the two databases. In this example, the student number provides the link.

For example, answering the question "Has John Doe ever taken Chemistry 101" might look like this (using the Paradox examples):

C	Sn	Name	
	<u>X</u>	John Doe	
D	Sn	Class	
	<u>X</u>	Chemistry 101	

There are actually two queries going on at the same time. In the C table, the selection criteria "John Doe" finds his record. In the D table, the selection criteria "Chemistry 101" finds records for that class. The X that appears in both query forms is an example element. The example element provides a method of linking the two query forms (and their underlying data tables) together. In this case, the query matches records in the two tables based on equivalent student numbers. Therefore, the result of the query is to find whether John Doe ever took Chemistry 101.

The second example, to find students who took Chemistry 101 in 1986 and received the grade of A, might look like this:

C	Sn	Name			
	<u>X</u>				
D	Sn	Class	Year	Grade	
	<u>X</u>	Chemistry 101	1986	A	

In this example, the selection criteria in the D database table find records for the desired class. The example elements X serve to find matching records (based again on the student numbers) in the C table.

You can see that using multiple tables for data storage makes using the database information a little more difficult, in that finding information generally involves querying more than a single table. But the benefits in efficient data storage outweigh this disadvantage.

---

## Index

---

Add'13  
Adding Data To Tables'11  
Adding Groups to Reports'38  
Calculations in Queries'25  
Changing Data in Tables'25  
ColumnSize'7  
Commands for Reports'43  
Copy'12  
Data Selection Queries'17  
Database Design'45  
Dates in Queries'21  
Delete'12  
Editing Tables'9  
Empty'12  
Examples in Queries'22  
Format'8  
Function Keys'1  
Image Settings for Tables'7  
Info'12  
KeepSet'7  
Literal Text and Field Masks'33  
Miscellaneous Commands'12  
Movement Keys'2  
Moving a Field'8  
Other Keys'3  
Paradox Menus'3  
Placing Totals'36  
Printing a Report'35  
Printing a Subset of Data'35  
Problems During Restructure'6  
Queries'17  
Range Queries'20  
Renaming'12  
Report Bands'32  
Reports'31  
Restructuring Tables'5  
Saving Queries'26  
Setting the Default Directory'4  
Summarizing a Group'40  
Summarizing Calculated Fields'42  
Summarizing Records'21  
Tables'4  
TableSize'7  
The Paradox Standard Report'31  
Undoing Editing Changes'10  
Using a Calculated Field'41  
Viewing Tables'6  
Zooming to Locate Data'10

---