PsiDat

Operating Manual

Index

INTRODUCTION
PsiDat is a database management system (DBMS) for the Psion Series 5/5mx/7 and Revo(Plus) range of computers. It should also run on all other EPOC devices although there may be issues with EPOC versions other than ER3 and ER5.

PsiDat is designed to open and edit OPL databases created by itself and other OPL applications. It can also open and read (but not write) databases created by the built-in EPOC Data application.

The program features flexible and intelligent import/export routines that include dBASE III/IV tables and delimited text. It can also output Lotus Notes Structured Text, Idealist Natural, and HTML.

Detailed information is provided as a standard helpfile that can be called from PsiDat's main menu. The helpfile will always reflect the latest version so it should be considered more authoritative than this manual which is designed to give an overview of PsiDat's facilities.

What is a Database ?

A *Database* is a file containing structured information. It may contain one or more tables.

A *Table* is a collection of similarly structured information. It will contain one or more records, all of which have the same basic layout.

A *Record* is a collection of data arranged in one or more fields.

A *Field* is a chunk of data in a known format. It has an associated *fieldname* to identify it.

A table is illustrated below :-

Fieldnames :-	**Product**	**Cost**	**Quantity**
Record 1	Eggs	0.10	100
Record 2	Bacon	0.20	200
Record 3	Beans	0.15	50

This table has three fields; 'Product', 'Cost', and 'Quantity'

The 'Product' field contains text (or string) data.
The 'Cost' field contains numeric data that can include decimals (ie. real numbers)
The 'Quantity' field contains numeric data that appears to just be made up of whole (integer) numbers, although there are only three records so this may not necessarily be the case depending on the intended use for the table.

The leftmost column headed 'Fieldnames:-' is not part of the table and won't be illustrated in subsequent examples in this document.

The built-in EPOC database application, Data, has a single table structure. This commonly referred to as a *flat file database*. Data is a powerful application in it's own right and for many purposes is more-than-adequate for the storage of data. It allows you to mix text, numerical, date, and *Rich Text* memos freely. You can even embed EPOC objects in the text.

It does have a few limitations however:-

- The printed output is very basic.
- It cannot perform calculations or summaries.
- You can only store one index (ordering definition) at a time.
- You can only search for exact matches.
- There is no facility for autofilling fields or using defaults when entering data.
- There can be a lot of repetitive typing when creating similar records.
- The data-entry is dialog-based making editing slow (this is particularly bad when editing long memos)
- It can only import/export text. Conversion to/from industry-standard formats requires the use of PsiWin and a PC or 3rd-party conversion routines.
- It can only embed certain types of objects.

It is worth looking at the more advanced form of database in conjunction with a common business problem.

Let's expand our sample database a bit to include all the fields that might be required to create a despatch note for some breakfast produce to our clients. We will need the following :-

Client
Contact
Address
Telephone
Order Number
Date
Discount
Item
Quantity
Description
Cost

Client	**Contact**	**Address**	**Telephone**	**Order No**	**Date**	**Discount**	**Item**	**Quantity**	**Description**	**Cost**
Smith Ltd	John Smith	Smithy Road	01234 567890	SL001	22/01/2001	10%	Eggs	100	Class A Eggs	0.10
Smith Ltd	John Smith	Smithy Road	01234 567890	SL001	22/01/2001	10%	Bacon	200	Bacon Rashers	0.20
Smith Ltd	John Smith	Smithy Road	01234 567890	SL001	22/01/2001	10%	Beans	50	15oz tin of Baked Beans	0.15
Smith Ltd	John Smith	Smithy Road	01234 567890	SL002	24/01/2001	10%				
Jones Ltd	Bob Jones	Keepupwith Lane	01234 678901	JL001	25/01/2001					
Smith Ltd	John Smith	Smithy Road	01234 567890	SL003	26/01/2001	10%				

We'll assume that only order SL001 has been completed at this time.

It can immediately be seen that there is very inefficient use of storage space in our database table. Our **Contact**, **Address**, **Order No**, **Date** and **Discount** are duplicated several times even for the same order number. We may also write the full **Description** and **Cost** of our **Item**s a great many times in the database. As this is done, errors and data input operator preferences may start to creep in. There must be a more efficient way of doing this, surely ?

There is - it's called a *relational database*.

Relational Databases

A database may contain more than one table. These tables may be unrelated or may have common fields that can be used to relate items in one table to those in another.

Here is our example laid out in relational form :-

Clients

Client	Contact	Address	Telephone
Smith Ltd	John Smith	Smithy Road	01234 567890
Jones Ltd	Bob Jones	Keepupwith Lane	01234 678901

Orders

Client	Order No	Date	Discount
Smith Ltd	SL001	22/01/2001	10%
Smith Ltd	SL002	24/01/2001	10%
Jones Ltd	JL001	25/01/2001	
Smith Ltd	SL003	26/01/2001	10%

OrderItems

Client	Order No	Item	Quantity
Smith Ltd	SL001	Eggs	100
Smith Ltd	SL001	Bacon	200
Smith Ltd	SL001	Beans	50

Items

Item	Description	Cost
Eggs	Class A Eggs	0.10
Bacon	Bacon Rashers	0.20
Beans	15 oz Tin Baked Beans	0.15

The four tables above share related data and colours and shadings have been used to illustrate these relationships.

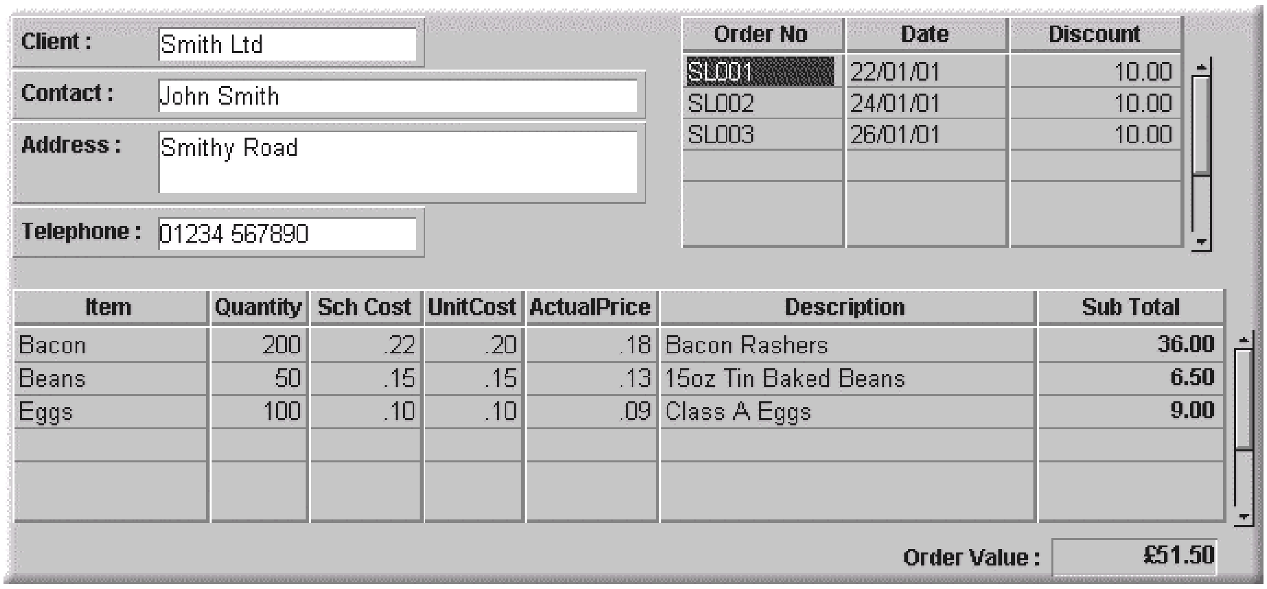
Data items generally appear only once, although the shaded fields are an exception. Although four tables have been used, the data organisation is better and takes up less space (45 cells instead of 66).

- For the small number of records used so far this wouldn't matter very much. However, if you had 100 clients, each with 100 orders containing 10 items each, then the **Contact, Address** and **Telephone** data would appear in 100 x 100 x 10 = 100,000 records instead of just once for each of your 100 clients. This would be very wasteful and require the information to be entered every time resulting in inconsistencies.

- On the other hand, you might not want to link the cost of the items directly to the ***Items*** table because if you wanted to change the cost of a rasher of bacon to 0.22, this would affect every item in the ***OrderItems*** table – even if it had already been ordered, delivered, and paid for. In this case, it might be better to duplicate the **Cost** field in the ***OrderItems*** table (we'll call it ***UnitCost***) and just copy over the value from the **Cost** field in the ***Items*** table when a new record is created.

These two points illustrate the pros and cons of building a relational database. For many business applications the pros vastly outweigh the cons.

Here is an example form from a PC relational database showing how these tables might be connected on an input screen :-



Client :	Smith Ltd
Contact :	John Smith
Address :	Smithy Road
Telephone :	01234 567890

Order No	Date	Discount
SL001	22/01/01	10.00
SL002	24/01/01	10.00
SL003	26/01/01	10.00

Item	Quantity	Sch Cost	UnitCost	ActualPrice	Description	Sub Total
Bacon	200	.22	.20	.18	Bacon Rashers	**36.00**
Beans	50	.15	.15	.13	15oz Tin Baked Beans	**6.50**
Eggs	100	.10	.10	.09	Class A Eggs	**9.00**

Order Value : £51.50

This shows all the items on order number SL001. An extra field **UnitCost** has been added to the ***OrderItems*** table. The form is programmed to copy the value called 'Sch Cost' (which is actually the linked **Cost** field in the ***Items*** table) into this field when it is left blank. A further column has been added called 'Actual Price' - this is just a calculation of **UnitCost** less the percentage discount indicated by the **Discount** field of the ***Orders*** table. Finally there is a 'Sub-Total' column – this is just a calculation of **Quantity** times 'ActualPrice'.

Notes:

- The **Order, Date, Discount** table at top right is a 'one-to-many' sub-table (actually ***Orders)*** of the ***Clients*** table. It also contains the field **Client** but it isn't necessary to show this because it is linked to the ***Clients*** information at top left.
- The **Item, Quantity, UnitCost** table at the bottom is a sub-table (actually ***OrderItems***) of the ***Orders*** table. The 'Sch Cost' heading is an *alias* of the real field **Cost** in the 'one-to-one' linked table ***Items***. It contains calculated virtual fields called 'ActualPrice' and 'Sub Total'
- In the bottom sub-table, only **Item, Quantity, UnitCost** can be directly edited in this form – the other columns are linked items or calculations.

If the user were to click below the item eggs and enter a new item, say 'Sausage', that was already in the ***Items*** table, the full Description and 'Sch Cost' would immediately appear with the appropriate information. The 'Sch Cost' would then be copied into the **UnitCost** field and the 'ActualCost' column would show this value less the **Discount**.

When the user enters a **Quantity** for the sausages, the ‘Sub Total’ and ‘Order Value’ values would be immediately calculated.

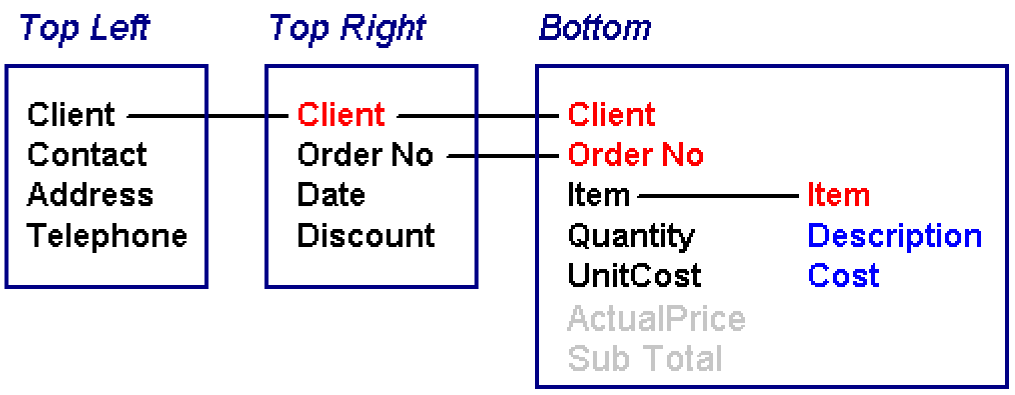
If the user decided to alter the **Discount** value, for order SL001, this would be immediately reflected in the ‘ActualPrice’, ‘Sub Total’ columns for all items, and the ‘Order Value’ would be recalculated.

A change in the ***Items*** table **Cost** for one of the items would result in the associated ‘Sch Cost’ value changing but this would not affect the **UnitCost** field because (in our form logic) this is only calculated if the value is empty, ie. when creating a new record.

If the user clicked on **Order No** SL002, all the items would disappear from the bottom sub-table and any for order SL002 would appear instead.

If the user clicked on the **Client** field at top left and used the database navigation tools to switch to another record, the ***Orders*** table would then show that client’s orders instead of those for ‘Smith Ltd’.

If this appears complicated, this sketch may help :-



Each box shows one of the three zones on the form.

The **black** text shows the fields you can edit in that portion of the form. The **red** text shows how sub-tables are linked to their parent table. The **blue** items can only be displayed, not edited, on this form. The **grey** items are not real fields at all; just temporary calculations.

What you can see in the top right table sub-table is restricted by which record is selected in the top left section. What you can see in the bottom sub-table is defined by which record (**Order No**) is selected in the top right sub-table.

Now consider one last option:-

The bottom table is currently linked by **Client** AND **Order No**. This allows more than one client to have the same order number without the orders getting mixed up. However, if we wished, we could allocate our own unique **Job Number** to every order. We could then link the bottom table to that unique number rather than **Client** and **Order No**.

If you’ve understood the concept up to now, you may be slightly disappointed to hear that PsiDat can’t actually do this at the moment, although it is a planned part of the development.

There are two reasons for this:-

- It involves a lot of work (!)
- It’s not immediately obvious how best to display this information on a small palmtop screen. Relational databases have only really come into their own with the availability of large screens with windowed forms; these don’t fit well into the palmtop computer environment. It will probably be necessary to provide a means of switching view to keep screen clutter to a minimum, rather than show the main and sub-tables simultaneously.

However, there are many powerful facilities and *semi-relational* features that can be used already, particularly for applications that require data-consistency and/or regular sharing of information with PC or mainframe databases.

A brief re-examination of the previous example will show that at the front-end, where a palmtop computer might be used, it may only be necessary to process the items on a single order and say whether these items have been fulfilled or not. We could cut the data down to a list containing **Order No**, **Item,** and **Quantity** and add a check item to say if this has been added to the order.

If we gave our operator a 2nd table containing the **Order No**, and **Client, Address, Contact,** and **Telephone** on a daily basis as the orders were added, and removed them once they were fulfilled, our front-line operatives would have all they required to despatch the customers’ orders and we could monitor the progress by receiving data from the palmtop computers. It may even be better to give the **Order, Item, Quantity** list to our warehouse personnel and give the **Order No, Client, Address, Contact, Telephone** list to our delivery driver. One deals with adding items to a series of order ‘packages’ whilst the other delivers those packages to various clients and has no need to know their individual contents.

In the next section, PsiDat’s current capabilities will be discussed in detail but hopefully this short introduction to databases will help to clarify where the program is heading and it’s current design philosophy.

Semi-Relational Features

PsiDat database files typically contain several tables. Some of these are named zPsiDat…and are created by the program itself. These are used to store variables connected with the table field properties, report layouts, SQL queries and various other parameters required to make the database behave in the desired way.

Lookups & Helps

The user will typically create one or more tables containing the desired fields for the information to be stored and may relate some to each other using the first text field.

There are two special uses for text fields that are important to understand in order to get the best out of PsiDat:-

1. A text field can be marked as a 'Lookup'
   This means that when the user tries to edit the field, a list box pops up and allows the user to pick the required string from a list stored in another table. The user is prevented from entering values that do not match the lookup table.

2. A text field can be marked as a 'Help'
   This initially appears to work in the same manner as a lookup. However, after clicking on the required item, or aborting (by pressing Esc or tapping outside the list), the field can still be edited. The user is not prevented from entering data that is not in the associated table.

Basic Lookup and Help tables can be generated by PsiDat just by setting a text field's 'Special Use' property to either. PsiDat will create a table called ***zPsiDatLookups***, if it doesn't already exist, and key entries to it by the tablename and fieldname. If the field is set as a 'Help' then creating new Help values can be done automatically whilst editing the main table. *(Tip: if you're going to use the built-in Lookups, set the field to be a 'Help' first until you've entered all the required Lookup values; then switch it to be a 'Lookup')*

Stand-alone tables can be used and are recommended when there are many items to choose from. These are especially useful when sharing data with a PC or mainframe application because it is possible to import the Lookup tables directly from dBASE tables or delimited text files. In this manner, it is possible to provide the end-user with a main table for their input, provide Lookups to make data-input quicker and more consistent, and limit the records produced to data fitting in to the desired ranges.

SQL Query Views

Nearly all databases allow you to search for data in a table to see if matches a text string or number. More advanced databases allow you search for more than one criteria at once.

A standardised form of searching for data has been defined and is the basis for *Structured Query Language*, or SQL as it is commonly known. SQL has grown into a complex language for accessing and manipulating database records. In some implementations it is possible to design entire databases using just SQL commands. [***In the author's opinion this approach is only recommended for masochists and should be avoided at all costs**.]*

A form of SQL is implemented in the EPOC DBMS but it cut down considerably from the full specification. The most severe limitations are listed below :-

- It is not possible to join (link) two tables together by making comparisons between common fields.
- It is not possible to directly compare one field with another or with a variable. Fields can only be compared to literal values.

  1. A string-literal is a character string enclosed in single quote characters ('). To include a single literal quote character (') in a string-literal, use two literal quote characters: ('').
  2. A numeric-literal is any sequence of characters which can be interpreted as a valid decimal integral or floating point number.
  3. A date-literal is a character string enclosed by the # character, which can be interpreted as a valid date.

This means that it is possible to create complex SQL expressions to filter out records based on fixed criteria but not based on comparisons between fields or across different tables.

The joining of tables for true relational joins between tables has to be carried out by the programmer manually. This involves building separate SQL queries within the program ‘on-the-fly’ and reopening the sub-table based on the field contents of the currently selected record in the main table.

The author has devised a workaround for the problem of comparing two fields within the same table. It has now been adopted for the Symbian OPL Knowledgebase:-

The method used to compare two fields is to add third ‘check’ field to hold the result of the comparison between the other two. The third field is recalculated everytime one of the other two changes and the query can then be performed by comparing this check field with a literal value.

This method can also be used to query the table based on the result of a complex equation between two or more fields.

To apply an SQL query in PsiDat a valid SQL search has to be performed. If this is successful then the user can then choose to ‘Apply SQL Filter’. This will restrict the displayed records to those matching the SQL criteria.

If a filtered record is modified it may drop out of the current restricted view. If this happens to the last record remaining record in the view then the ‘Apply SQL Filter’ is toggled off. However, if the table is currently using an index, it may not drop out of view immediately.

Tutorial

The best way to learn an application is to create something in it. This chapter will take you through the process of constructing a database to monitor business expenses and produce expense claim reports. The database described is designed for UK VAT but could easily be adapted for other countries.

Specification

We want to produce a means of storing expenses that includes the following information:-

1. The date of the expense
2. A category for the expense (eg. Travel, Subsistence, Entertaining etc.)
3. A cost code centre describing which project or part of the organisation eventually pays for the expense.
4. The amount.
5. The proportion of the amount that is tax (in this case UK VAT)
6. An ID so that one group of expenses can be isolated from another (eg. Expense form number)

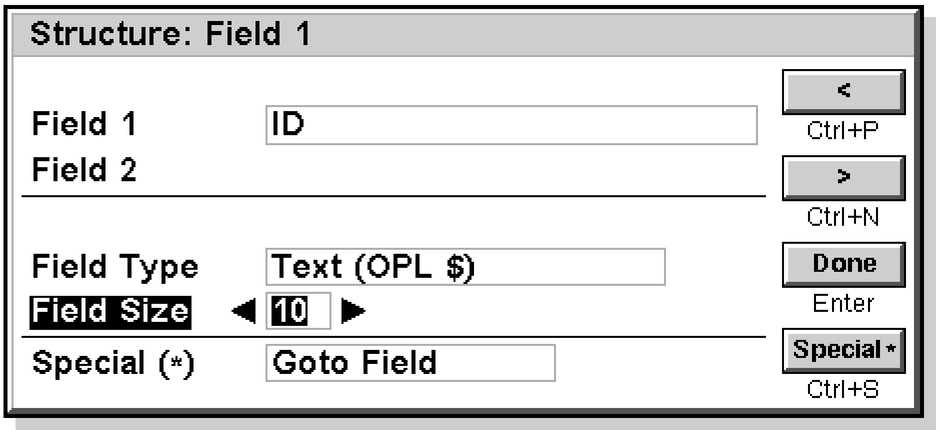
We need to include :-

1. The possibility that VAT may not apply to all items.
2. A system for adding up all the expenses.
3. A report that could be used as the basis for an expense claim form (or at least to assist with filling in one)

Table Definition

Start PsiDat and select ‘Create New Table’ <Ctrl N>

Define the first field as a text field of length 10 with the name ‘ID’ :-


Structure: Field 1

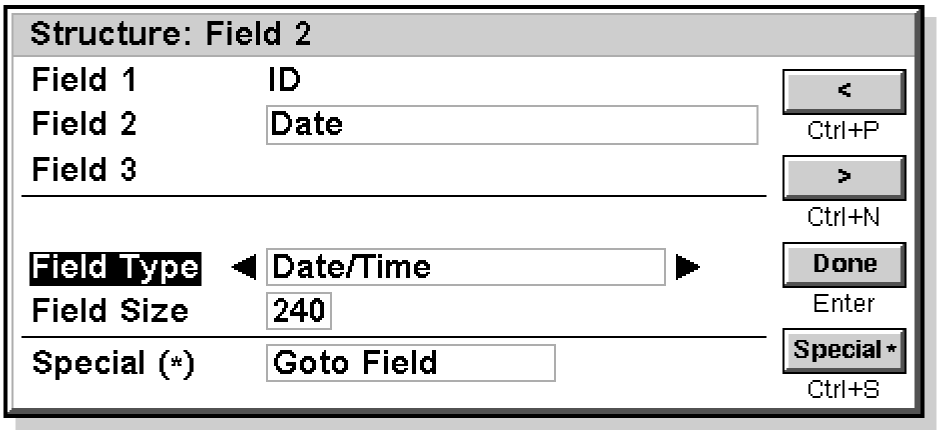
Field 1 ID
Field 2

Field Type Text (OPL $)
Field Size ◀ 10 ▶

Special (*) Goto Field

Tap the ‘>’ button to move to Field 2

Then define the 2nd field as a DateTime type with the name ‘Date’ *(Don’t worry about the field length – this is ignored unless it is a text field type).*


Structure: Field 2

Field 1	ID
Field 2	Date
Field 3	
Field Type	Date/Time
Field Size	240
Special (*)	Goto Field

Carry on doing this until you have defined the following :-

Name	Field Type	Field Size
ID	Text (OPL $)	10
Date	DateTime	*Ignored*
Category	Text (OPL $)	20
Details	Text (OPL $)	50
ApplyVAT	Bit	*Ignored*
Amount	Double Float (OPL Real)	*Ignored*
VAT	Double Float (OPL Real)	*Ignored*
CostCode	Text (OPL $)	10
Comments	LongText8 (Memo)	*Ignored*

Then tap on the 'Done' button and enter the following :-

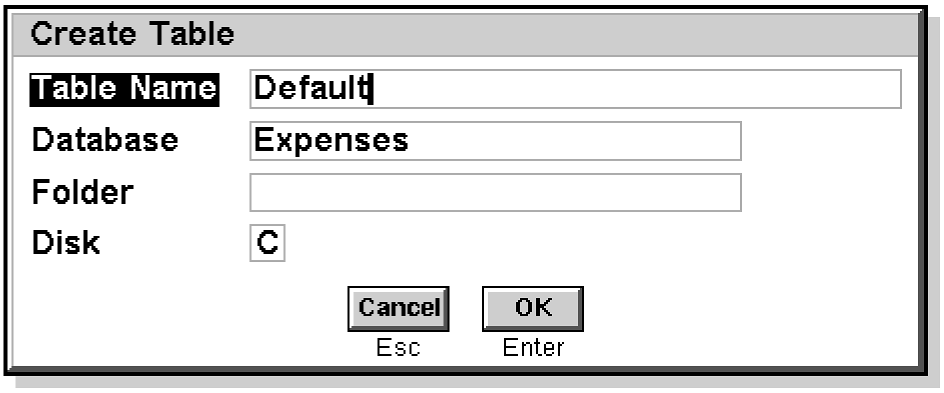
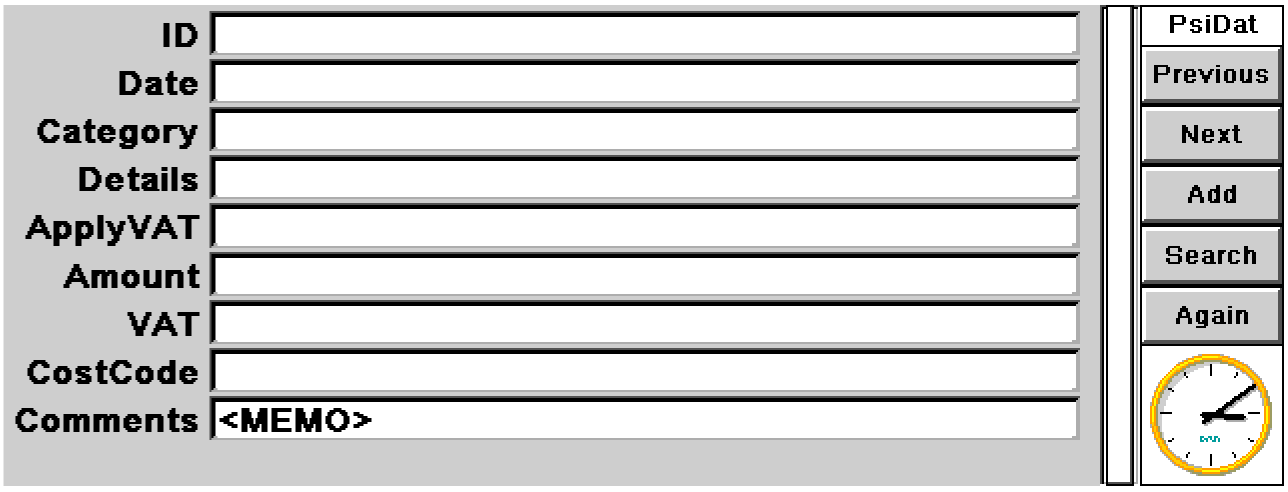

Create Table

Table Name	Default
Database	Expenses
Folder	
Disk	C

Notice that when you create a PsiDat table you have to enter both the name of the Database (the filename) and a table name.

Select the 'Open…' option from the main menu and click on 'OK' to load the Expenses Database. You should see something that looks like this :-



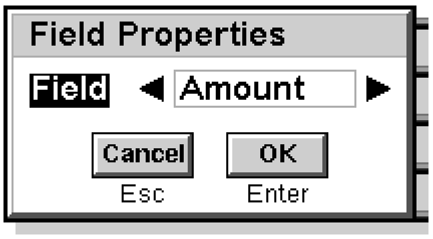
If you can't see the Toolbar, press 'T' to display it.

Congratulations ! – you've created a working database table.

Enhancements

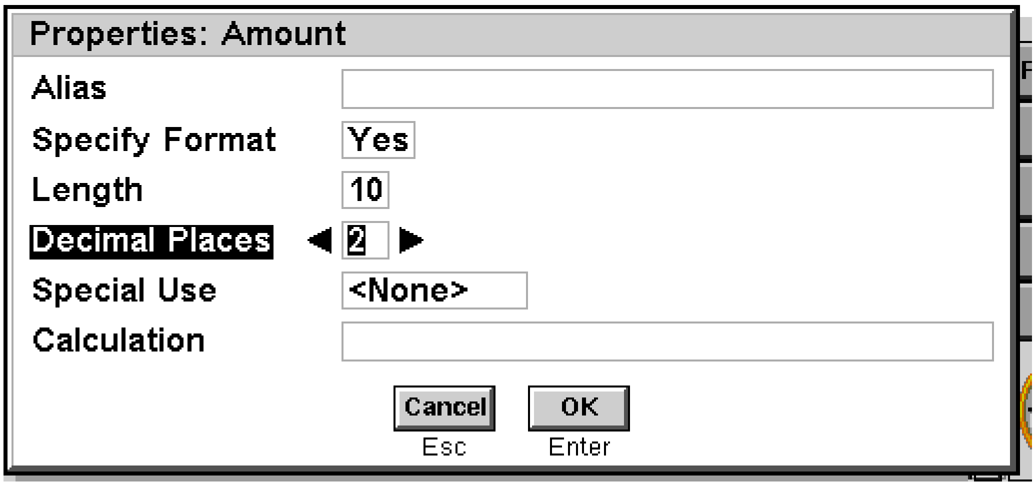
It is desirable to restrict the format of the **Amount** and **VAT** fields. Select the menu and under 'Options', choose 'Field Properties'.

Select the **Amount** field :-


Field Properties

Field ◀ Amount ▶

Now define the properties for the **Amount** field :-

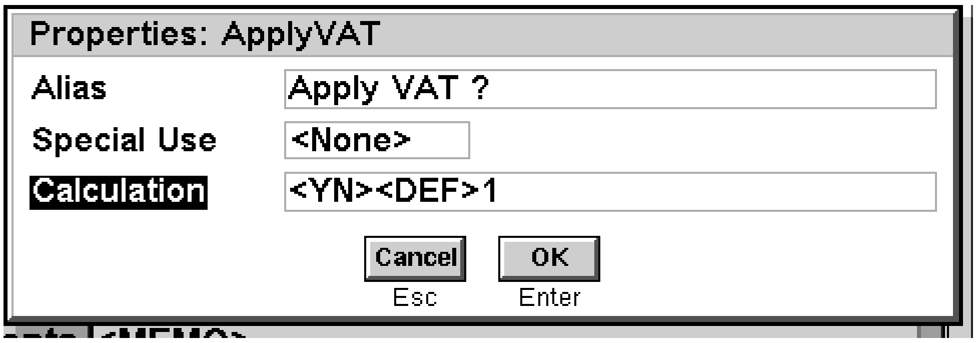

Properties: Amount

Alias	
Specify Format	Yes
Length	10
Decimal Places	◀ 2 ▶
Special Use	<None>
Calculation	

This will force the numbers in the **Amount** field to display with two decimal places.

Do the same for the **VAT** field.

Now change the properties for the **ApplyVAT** field to :-

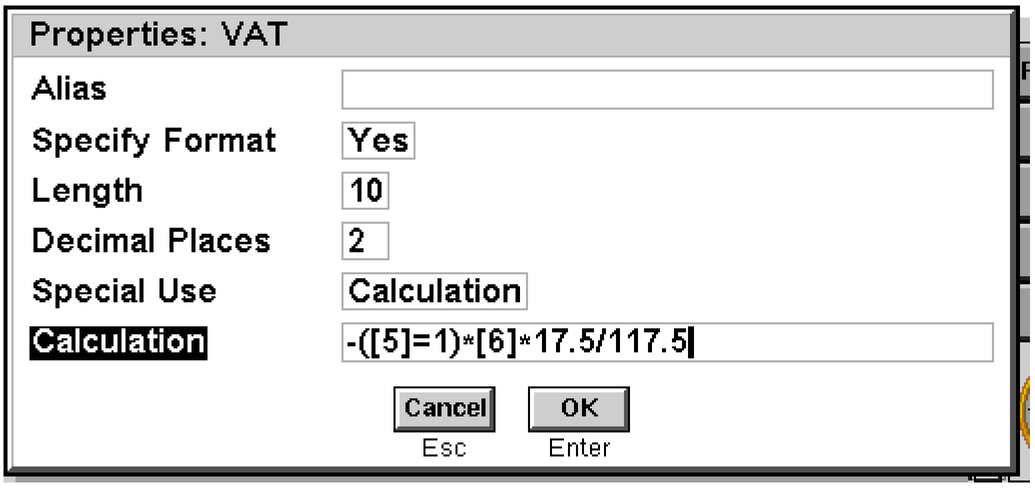


Properties: ApplyVAT	
Alias	Apply VAT ?
Special Use	<None>
Calculation	<YN><DEF>1

There are three features being applied here:-

1. *The 'Alias' option improves the appearance of the field label in the display.*
2. *2. The* **<YN>** *in the 'Calculation' option is a field modifier telling PsiDat to display 0 as 'No' and 1 as 'Yes'. NB: although this modifier goes in the 'Calculation' text, this is NOT a calculation so the 'Special Use' is left as '<None>'*
3. *The* **<DEF>1** *at the end of the 'Calculation' option is a field modifier telling PsiDat to use the value 1 as the default when a new record is created.*

It would be desirable to calculate the VAT automatically so look at the **VAT** field properties again, change the 'Special Use' to 'Calculation' and enter the following formula :-

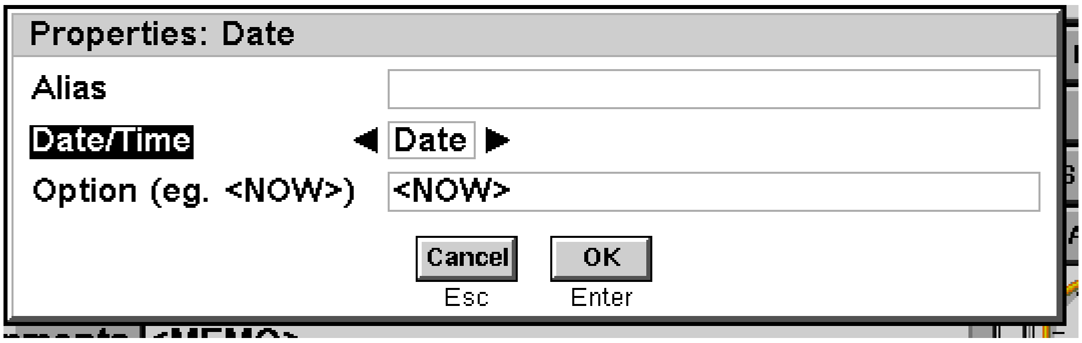


Properties: VAT	
Alias	
Specify Format	Yes
Length	10
Decimal Places	2
Special Use	Calculation
Calculation	-([5]=1)*[6]*17.5/117.5

Here we are telling PsiDat to perform a calculation based on the contents of field [5] and field [6]. The **-([5]=1)** *portion is a comparison between the* **ApplyVAT** *field and the value 1. Strictly you could just use* **[5]** *here because, as a Bit field, it can only be 0 or 1. However, this is a useful way of showing how to make a value zero if another field doesn't match a variable, or 1*something if it does. NB: the value 'True' is strictly –1 and this is the reason why there is a minus sign in front of the* **([5]=1)** *term.*
The result of this comparison (0 or 1) is then multiplied by the contents of field [6] (ie. the **Amount***) times 17.5/117.5*

If you prefer, try changing the formula to **[5]*[6]*17.5/117.5** *just to confirm this gives the same result.*

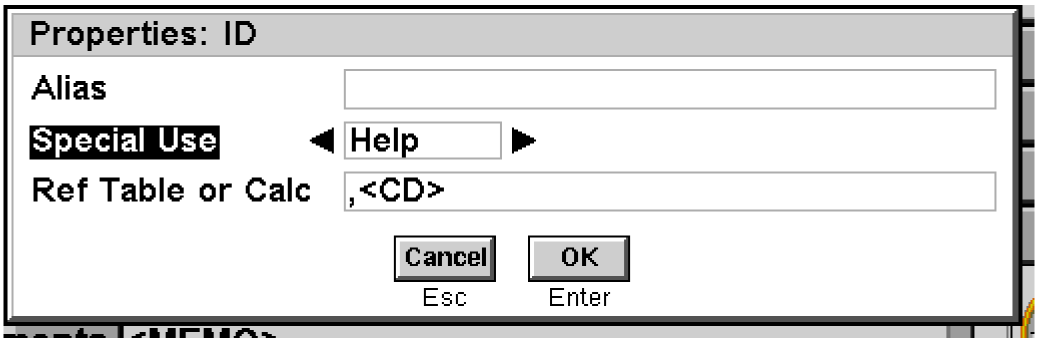
It would be desirable for the **Date** field to default to the current date whenever a new record is created. Change the field properties for this field to :-


Properties: Date

Alias	
Date/Time	◀ Date ▶
Option (eg. <NOW>)	<NOW>

Note that the 'Date/Time' option has been changed to just 'Date' because we're not interested in the time aspect (although this information is still recorded)

Change the **ID** field properties by adding the **<CD>** modifier and changing the 'Special Use' option to 'Help' :-

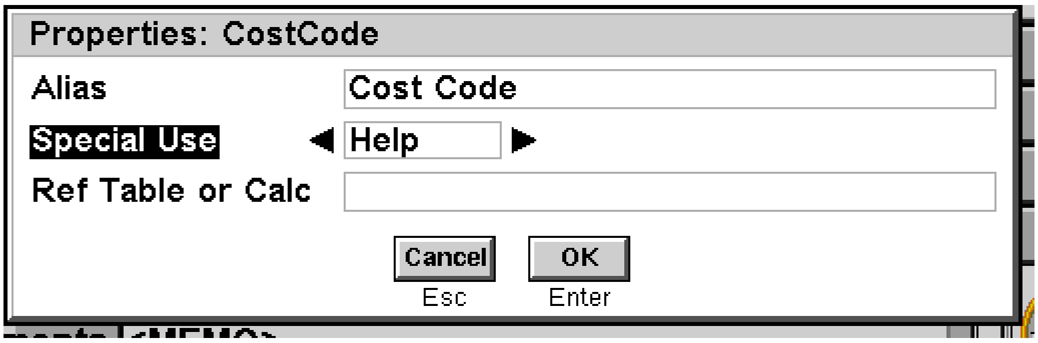

Properties: ID

Alias	
Special Use	◀ Help ▶
Ref Table or Calc	,<CD>

This cause the value in the current record to be copied down when you create a new record. This allows expenses to be added gradually to an expense claim without having to type in the expense ID every time. It also allows the ID number to be stored as a help text so that they are easier to keep track of.
NB: The comma before the <CD> option is crucial when you define a Help or Lookup – if you don't include this comma PsiDat will think you are trying to use a Lookup table called '<CD>'.

Add the 'Help' 'Special Use' option to the **Category** field as well but without the **<CD>** modifier.

Add the 'Help' 'Special Use' option to the **CostCode** field as well and change it's alias to 'Cost Code' :-

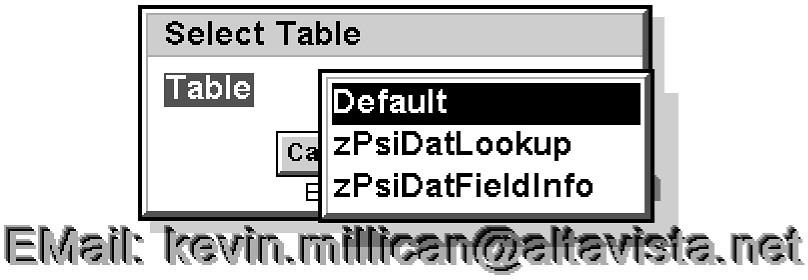

Properties: CostCode

Alias	Cost Code
Special Use	◀ Help ▶
Ref Table or Calc	

You may want to add the copy-down modifier **<CD>** *as well (don't forget the comma !) – it's up to you, although you can always change it later.*

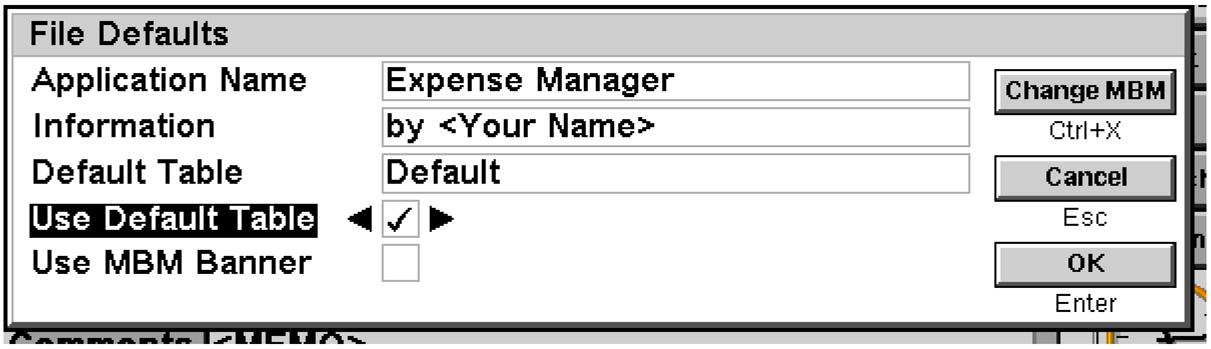
We will modify these field properties again when we look at the reporting side of things. For now, close the table (to save the field properties) and then reopen the Expenses Database.

You'll notice that you now get asked to select a table. Tap the table field and you will observe that there are now three tables in the Database :-



PsiDat has created two to hold the Lookup/Help texts and to store the Field Property modifications that you made.

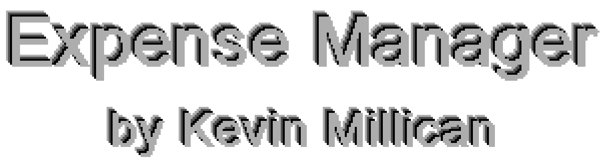
For most databases, you'll probably want the same table to be displayed first every time it is opened. Open the 'Default' table and then select 'Default Table' from the 'Options' menu. Complete the Dialog as follows :-



File Defaults

Application Name	Expense Manager
Information	by <Your Name>
Default Table	Default
Use Default Table	✓
Use MBM Banner	

Close and reopen the table and you will now see the following banner :-



…with your name of course !

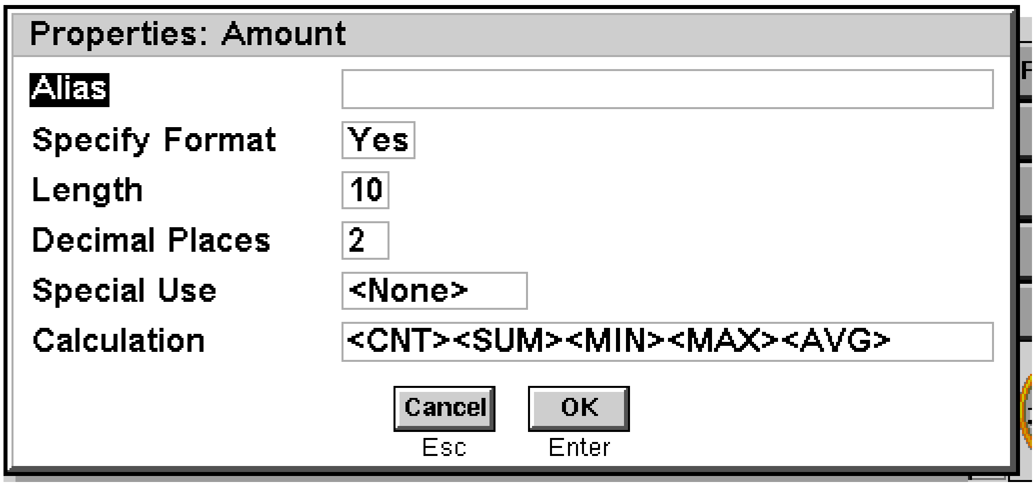
You should note that this is <u>your</u> application. Obviously it runs under the PsiDat DBMS, but as PsiDat is freeware you may, without royalty or any other acknowledgement to the author, distribute the Expense Manager application, or any others you develop, freely or commercially. The only conditions on this are that you do not modify any SIS files produced by the author and that the author will not accept any responsibility for consequences arising out of using the PsiDat software – use at your own risk. By using PsiDat you imply your full acceptance of these conditions.

Of course, there are probably quite a few PsiDat Expense manager applications out there already !

Summary Fields

As this is an expenses application, it's likely that it will need to summarise the expenses and provide totals for the **Amount**s and **VAT**. It would also be useful to count the number of **Amount**s.

Examine the field properties for the **Amount** field and add **`<CNT><SUM>`** to the calculation option (but leave the 'Special Use' option set to '<None>')

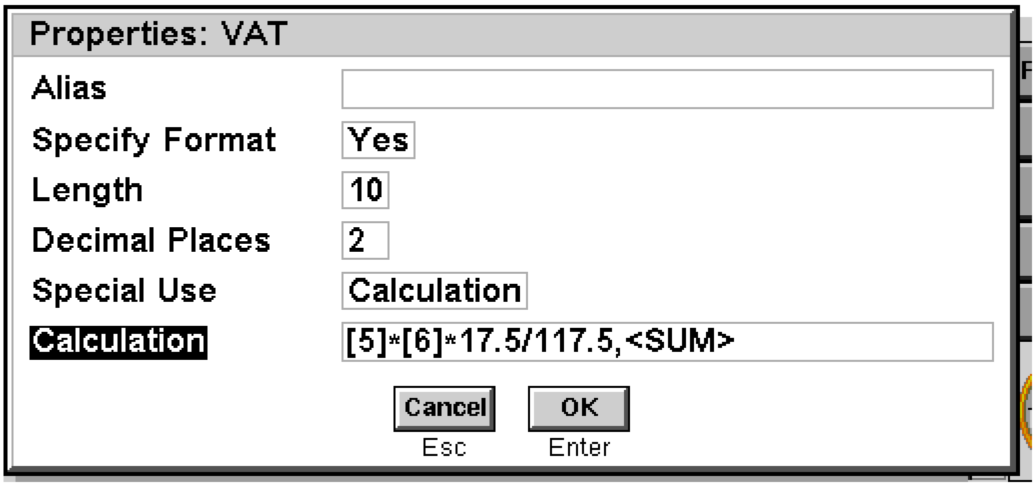


Properties: Amount

Alias	
Specify Format	Yes
Length	10
Decimal Places	2
Special Use	<None>
Calculation	<CNT><SUM><MIN><MAX><AVG>

In the screenshot above, all five summary field modifiers have been added; you may like to try all five and then cut them back to the two later or even leave them all in.

Now examine the field properties for the **VAT** field and add **`<SUM>`** to the calculation option after a comma:-



Properties: VAT

Alias	
Specify Format	Yes
Length	10
Decimal Places	2
Special Use	Calculation
Calculation	[5]*[6]*17.5/117.5,<SUM>

The comma tells PsiDat that the **`<SUM>`** *modifier is not part of the calculation equation. (Note: the more usual method of using the* `[5]` *Bit field result in an equation is shown here).*

If you have entered a few test records then selecting the 'Show Summary Values' feature from the 'View' part of the main menu will display the totals.

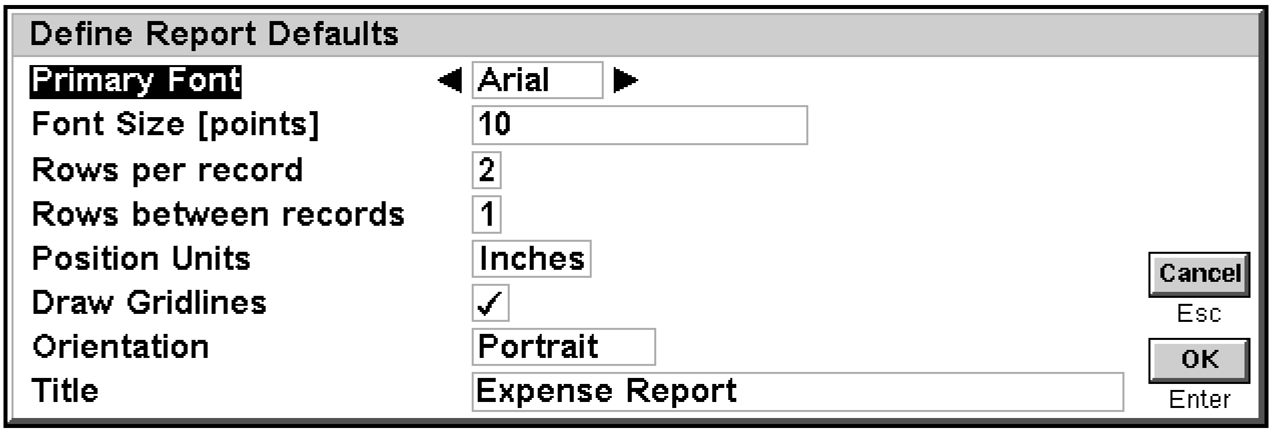
Reports

PsiDat can store as many different report formats as you need to provide suitable printed reports. At present these reports are limited to a row-based report similar to that produced by a spreadsheet. However, each record is not restricted to a single row; up to 8 rows can be used and each record can also be separated by 1 or 2 blank rows if required.

This section shows how to create a basic report for the Expenses Database.

Select the 'Define Report' option from the main menu (it's under 'File | Reports>').

Fill in the defaults dialog as follows :-

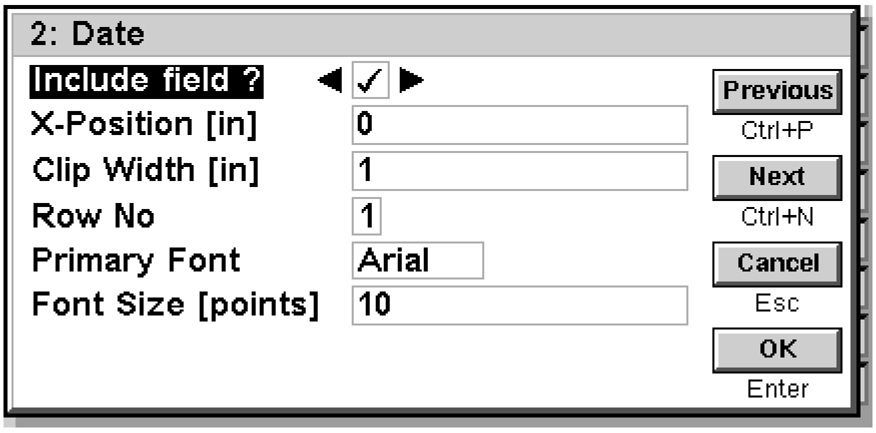


Define Report Defaults

Primary Font	Arial
Font Size [points]	10
Rows per record	2
Rows between records	1
Position Units	Inches
Draw Gridlines	✓
Orientation	Portrait
Title	Expense Report

This dialog defines the basic properties for the report and is the basic of the line spacing. However, the font properties can be varied for each field.

You then define position and font properties for each of the fields. The first one, **ID**, won't be used so leave this unticked and press the 'Next' button to move to the 2nd field. Tick the 'Include Field' option, leave the 'X-Position' at zero (ie. the left margin) and set the 'Clip Width' to 1 :-



2: Date

Include field ?	✓
X-Position [in]	0
Clip Width [in]	1
Row No	1
Primary Font	Arial
Font Size [points]	10

Now move through the fields using the ‘Next’ button and complete the dialog for each as follows :-

Fieldname	Include Field	X-Position	Clip Width	Row No	Primary Font	Font Size
ID						
Date	Ticked	0	1	1	Arial	10
Category	Ticked	6.2	0.9	2	Arial	8
Details	Ticked	2.2	4	1	Arial	10
ApplyVAT						
Amount	Ticked	1	0.6	1	Arial	10
VAT	Ticked	1.6	0.6	1	Arial	10
CostCode	Ticked	6.2	0.9	1	Arial	10
Comments	Ticked	2.2	4	2	Roman	8

When you are certain all the details have been entered correctly, click on ‘OK’ and then select ‘Save Report’. Give the report a name, eg. ‘Portrait’ and click on ‘OK’ to save the definition.

You can now print a fairly detailed report provided that your EPOC device is setup correctly to print, either directly or via the PsiWin link.

If the comments field prints out as ‘<MEMO>’, go to the main menu and select Options | General... Then tick the ‘Show Memo Text’ field.

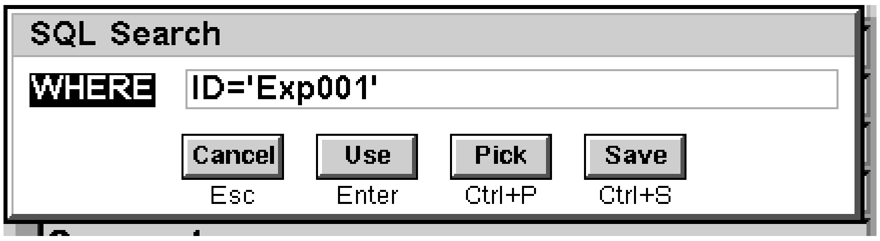
Selecting Information

The Expenses Database is now complete.

To select records for a particular expense claim or report, press Q to call up the SQL Search dialog.

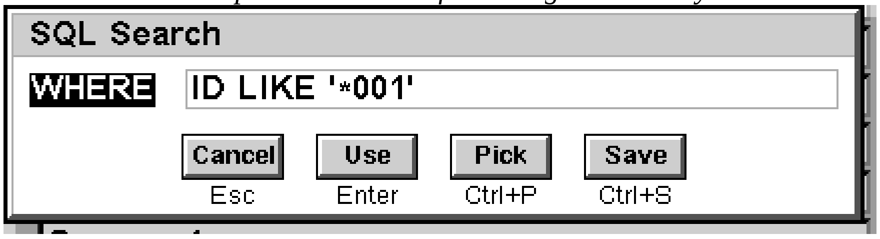
Suppose that the **ID**s for your claims are identified Exp001, Exp002 etc. and that you have about 30 expense items altogether; 8 on Exp001, 15 on Exp002, and 7 on Exp003. Obviously you will want to view the summary for, or print one particular report without including all the other records.

In the ‘WHERE’ option of the dialog enter:-



WHERE ID='Exp001'

Note the use of single quotes to surround the desired text. If you were searching for a date you would use # signs to enclose the desired date. Numerical values are not enclosed in anything.
If you wish, you can use the LIKE operator instead of the = sign. This lets you use wildcards :-



WHERE ID LIKE '*001'

If this matches any record, pressing the Enter key will take you to the first matching expense item.

You can then press Shift-Q to filter the records. If this is successful the vertical record position indicator will change from white to grey.

When the table has the SQL filter applied in this manner, summaries and reports will only include the matching records.

The SQL filter will also affect exports, copying all records to the clipboard, 'Restructure', and 'Save as' operations.

Some edit operations will affect the validity of the current record within the filter. For example, when you add a new record, there is no guarantee that you will satisfy the filter criteria. In these instances, PsiDat will disable the filter and revert to showing all records. You can reapply the filter by pressing Shift-Q or selecting the 'Apply SQL as Filter' option from the main menu.

At any time you can see whether the filter is applied by looking at the colour of the record position bar.

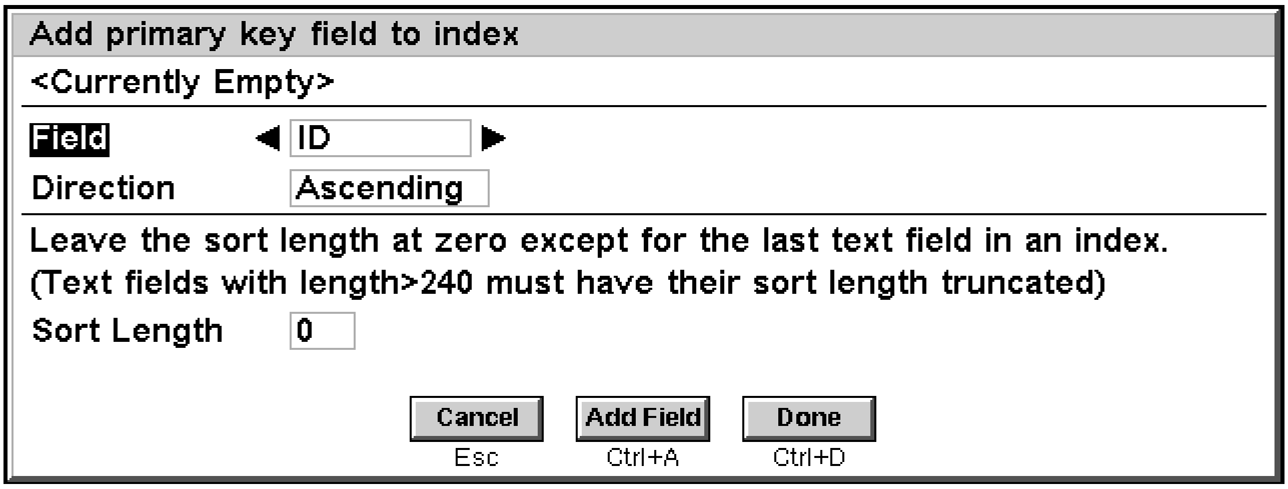
As you use more filters, you will probably find it useful to save successful ones so that these can picked from a list by pressing the 'Pick' button in the SQL Search Dialog. If you want to delete one from the picklist, just save it again and PsiDat will ask if you want it deleted.

Indexes

As the Expenses Database gets filled up, you may wish that you could order the records in a more sensible fashion. It would be useful to store the records generally in date order, but give the **ID** prority. To do this it is necessary to add an index. You can have more than one index for each table in your Database but each index adds to the overall size of the table.

Close the Database (Ctrl-X) to return to the PsiDat initial design menu.

Select Structure | Add Index… from the menu and pick the Expenses Database and Default table. The following dialog box is displayed :-


Add primary key field to index
<Currently Empty>

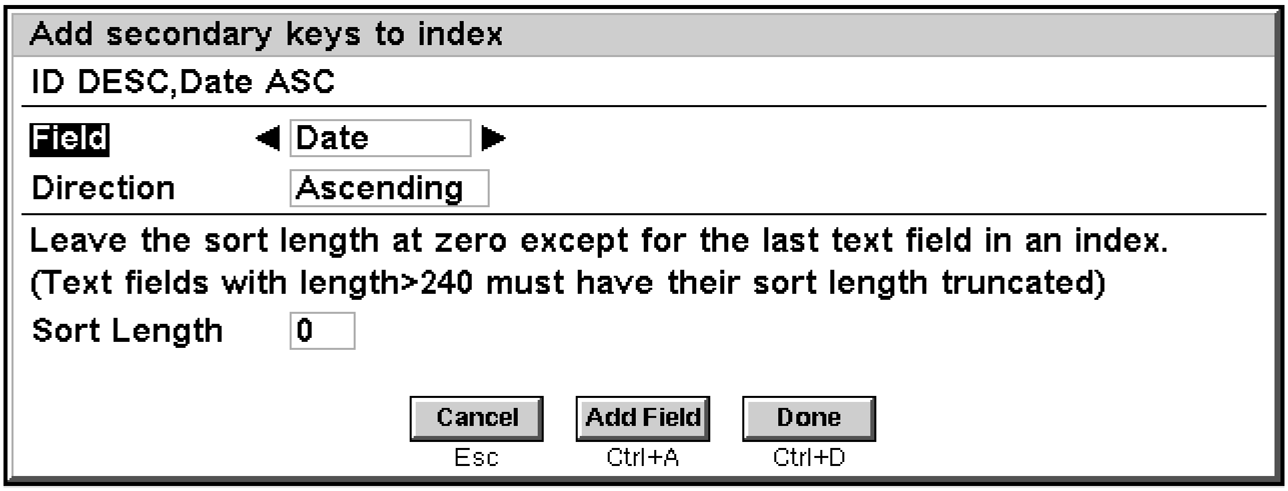
Field ◀ ID ▶
Direction Ascending

Leave the sort length at zero except for the last text field in an index.
(Text fields with length>240 must have their sort length truncated)

Sort Length 0

Change the direction option to 'Descending' and click the 'Add Field' button.

Then pick the **Date** field and change the direction option to ‘Ascending’. Click the ‘Add Field’ button again and you should see the following :-


Add secondary keys to index

ID DESC,Date ASC

Field ◀ Date ▶

Direction Ascending

Leave the sort length at zero except for the last text field in an index.
(Text fields with length>240 must have their sort length truncated)

Sort Length 0

At this point you can click the ‘Done’ button to save the index. Give it a name that means something, eg. IDdDate, and click on ‘OK’ to save. *Do not tick the ‘Unique’ option – this would prevent you from having more than one record with the same ID and date (though this can be useful for many applications).*

Now when you open the Expenses Database it will always show the topmost **ID** records first and within each group of records with the same ID, the records will be sorted by date.

Index Comments

There are a couple of points to note when creating and using indexes:-

- You can only use numeric, text, and datetime fields to build the index.
- The record number can become inaccurate when using a table with an index. This occurs when a record’s index key fields are edited so that its position in the view changes. Accuracy can be restored by pressing F to move to the first record or using the ‘Goto Record’ feature.
- Text fields up to 240 characters in length can be added to an index’s keys. If the text length exceeds 240 characters, it will have to be truncated. This truncated key must always be the last key in the index and so there can only be one. Therefore, as soon as you add a truncated field to the list of keys, you will see the save index dialog (you can’t add any more keys after this).
  This is the reason why the default text field length is set at 240 characters.
  You can truncate shorter text fields (eg. to save space – 20 characters is usually adequate) but the rule is the same; only one truncated key per index and it must be the last key in the index. A little care over the design of the initial table structure can pay dividends here.

Reference

Field Modifiers

A field modifier is a flag put in the 'special use' or 'calculation' line of the field properties dialog. The following field modifiers are supported:-

Modifier	Comments
Summary Operators	
<CNT>	The number of records counted
<SUM>	The sum total
<AVG>	The average
<MIN>	The minimum value (will be affected by the ‘Count Zero Values’ Option in the General Preferences)
<MAX>	The maximum value in the view
New Record Autofills	
<CD>	Copy down the current record field data to the new record
<+1>	Add one to the value in the current record and use this new value in the new record
<DEF>	Default value: everything after the ‘>’ character is the default value inserted when a new record is created
<NOW>	Put the current datetime value in when a new record is created.
Recalculation	
<MOD>	Update this field to the current datetime when a recalculation occurs (effectively this is a ‘DateTime Modified’)
Display/Input	
<YN>	Yes/No : for Bit fields only
<TF>	True/False : for Bit fields only
<DMS1>	Deg,Min,Sec format for Lat/Long numbers: DD.DDDDD°
<DMS2>	Deg,Min,Sec format for Lat/Long numbers: DD° MM.MM’
<DMS3>	Deg,Min,Sec format for Lat/Long numbers: DD° MM.MMM’
<DMS4>	Deg,Min,Sec format for Lat/Long numbers: DD° MM’ SS”
<DMS5>	Deg,Min,Sec format for Lat/Long numbers: DD° MM’ SS.S”
<UC>	Display text in uppercase
<LC>	Display text in lowercase
Information	
<INF>	Show the data in this field as part of the Record Info in addition to ‘# of n’ : up to three fields can be flagged in this way and it can help when navigating records with many fields. NB – this option is only activated when the table is opened in order to keep the speed up.

Navigation and Useful Keypresses

Keys	Action	Alternative
Left Arrow	Previous Record	Toolbar button / Menu
Right Arrow	Next Record	Toolbar button / Menu
F	First Record	Menu
L	Last Record	Menu
G	Goto a specific record number	Menu / Tap on Record Position Bar
Up Arrow	Move up the fields	Scrollbar
Down Arrow	Move down the fields	Scrollbar
Pg Up	Move up the fields one page	Scrollbar
Pg Dn	Move down the fields one page	Scrollbar
Home	Move to the top of the fields	Scrollbar
End	Move to the bottom of the fields	Scrollbar
A	Add a new record	Toolbar button / Menu
Del	Delete the current record	Menu
S	Basic text search	Toolbar button / Menu
Q	SQL search	Menu
A	Search again (by either method)	Toolbar button / Menu
Shift Q	Apply SQL as a filter	Menu
R	Recalculate	Menu / or edit field (with auto recalc on)
T	Show/Hide Toolbar	Menu
Shift T	Show/Hide Scrollbars	Menu
Ctrl O	Switch Table	Menu
Ctrl X	Close Database (back to design menu)	Menu
Ctrl E	Close PsiDat	Menu / System Task List
H	Display the online Help	Menu

Most of the menu shortcuts are available using the Ctrl key as reported in the menu and will also function on their own. Exceptions to this include: closing the program (Ctrl-E), closing a database (Ctrl-X), changing the table (Ctrl-O) and the printing functions.

Copyright

Disclaimer

Contact

The preferred method of contacting the author is by email at :-

kevin.millican@altavista.net

or,

kevin@millican.net