Introduction 2
Licensing 2
Introduction 3
File 4
Data 4
Procedure 4
Program 5
Database 5
Tabfile 6
Settings 6
Help 6
Database 7
Utilities 8
Master 8
Tabfiles 8
VisualPQL 9
SirSQL 9
PQLFORMS 10
Building an Application 10
COMPANY Database 11
VisualPQL 12
A first program 13
Steps to run a first program 13
VisualPQL Syntax 15
Example VisualPQL programs 16
More VisualPQL 18
Logical Conditions 18
Database Access 18
Accessing Data from the Database 19
Database 21
Creating a new database 21
Example Database: 21
Maintaining Databases 23
Safeguarding Databases 23
Writing Data 24
Procedures 25

Introduction

SIR/XS is a comprehensive database management and application development system. It helps you to organise and store data, to process and manage that data and to produce reports and other outputs. You can use SIR/XS to build complete applications.

If you have never used SIR/XS before, this is the right place to start. The material here is an introduction to SIR/XS which is intended to help you understand the software and to start to do some useful work.

Licensing

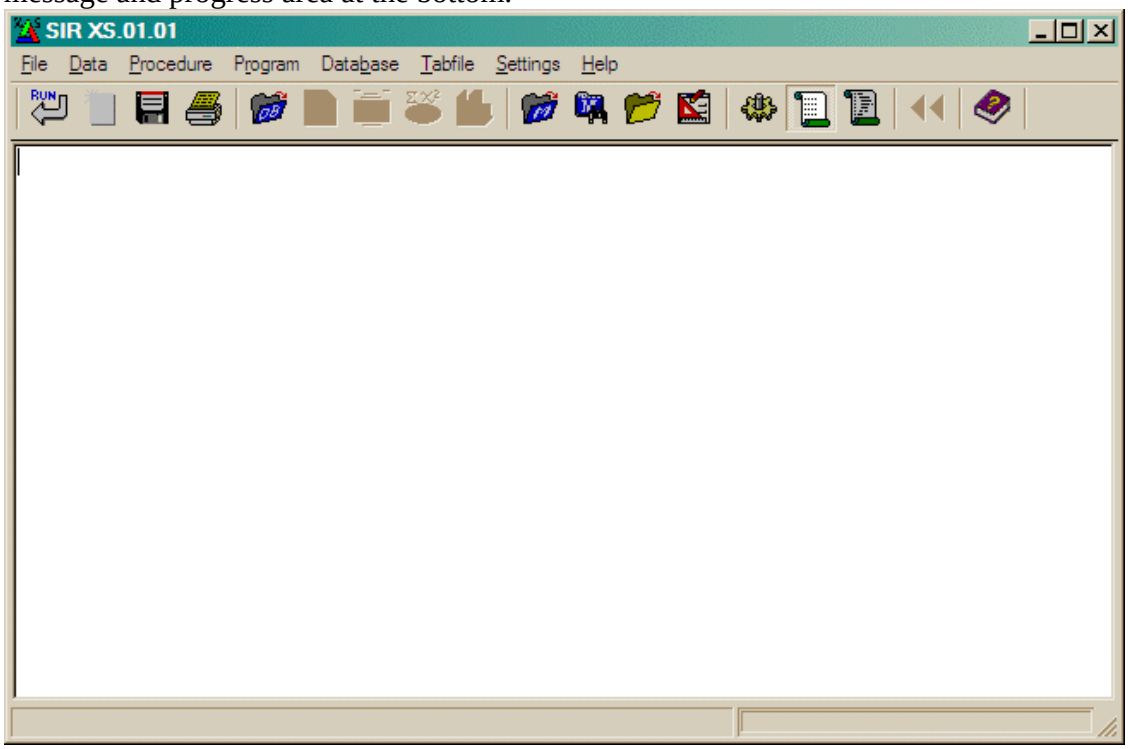
SIR/XS has a licence manager and you must have the appropriate licence code in order to use the software. This is checked each time you start the system. If you do not have a licence, please contact SIR to register at:

- **Email:** info@sir.com.au
- **Web:** http://www.sir.com.au

If your licence is about to expire, the system warns you and allows you a period of grace to get a new licence.

Introduction

When you start SIR/XS, you see the main SIR/XS window. This consists of a title bar, a main menu and toolbar at the top, a large scrollable output area in the middle and a message and progress area at the bottom.



The Default Main Menu

The title bar displays the name of the application, the default database (if any) and, if concurrent database access is being used, the name of the controlling master process.

The menu (and subsequent sub-menus and dialogs) is the main way of interacting with SIR/XS. (Note. These menus and dialogs are all written in VisualPQL and the source of all programs is included on the system procedure file and can be altered or replaced by custom built menus.)

Select a choice from any menu with the arrow keys, with a mouse or other pointing device or with underlined "hot keys" (use Alt-letter in the standard way) or by any indicated control key. Some of the frequently used items can also be activated from the toolbar.

Return to a previous menu by pressing `Esc`.

Menu items may pull down further menus (indicated with a small arrow), may display a dialog which you fill in or may take an immediate action. Some actions may generate

output. This is displayed in an output window which can be scrolled (up/down/left/right) and is used for remarks, messages and screen listings. The amount of remarks, commands and other internally generated messages can be controlled through session options. You can select and cut from the output window and can print it, save it or clear it. If your session produces more output than it can hold, earliest lines are discarded.

When you start SIR/XS, you may get a 'Welcome to SIR/XS' dialog asking what you would like to do. This allows you to create a new database or to connect to an existing database. Press Close to use SIR/XS without a database. Check "Don't show me this again" to suppress this. It can be set to display again in the preferences dialog.

The standard SIR/XS interface has the following menu structure:

File

The file menu can connect to database files and controls the contents of the session window.

- New... Creates a new SIR/XS Database.
- Open... Opens an existing SIR/XS Database or other file.
- Save Session... Saves current settings, globals and buffers.
- Restore Session... Restores settings, globals and buffers.
- Clear Output Clears the output window.
- Save Output... Saves the output window as a text file.
- Find in Output... Search the output for a string.
- Select All... Select all output.
- Print... Prints the output window.
- DBMS Command... Allows you to enter SIR/XS commands or run commands typed into the main window.
- Exit Exits the system.

Data

The data menu provides utilities to view and update the data in the database.

- SpreadSheet... Displays data in spreadsheet style.
- Forms... Creates and executes SIRForms data entry screens.
- File Dump... Creates text file of data (input for File Input).
- File Input... Updates database from text file.
- List File... Creates detailed report of database contents.

Procedure

The procedure menu allows you to produce outputs specifying various selection criteria.

- Export Data... Exports SIR/XS data to other packages.

- Report... Produces reports.
- Tabulation... Produces cross tabulations.
- Statistics... Produces descriptive statistics on a single variable.
- Graph... Produces 2D or 3D graphs using two or three variables.
- More Procedures...Run previously saved user procedures.
- Open... Open a previously saved procedure.
- Preferences... Change some of the procedure output settings.

Program

The program menu allows you to manage your VisualPQL programs and other text members and files.

- Members... Edit, run and manage procedure file members.
- Search Members... Search members by attribute or contents.
- Compare Procfiles... Compare two procfiles by contents, name or other attributes.
- Files... Edit or run files
- Dialog Painter... Invoke the dialog painter to update dialog based programs.
- PQLForms Painter... Creates and executes PQLForms data entry screens.
- Debug... Start the GUI debugger on a previously debug-compiled routine.

Database

The database menu allows you to manage databases. It provides backup and restore facilities and contains dialogs for modifying and displaying the structure of a database.

- New... Creates a new SIR/XS Database. (Same as File/New...).
- Open... Opens an existing SIR/XS Database or other file. (Same as File/Open...).
- Control Databases...Controls connections to SIR/XS databases.
- Export... Save the database in a machine independent text format.
- Unload... Save the whole database or a subset in internal format.
- Journal Upload... Save the journal as a machine independent text file.
- Journal Restore... Apply journal updates to earlier version of database from journal in internal format.
- Download Journal... Apply journal updates to earlier version of database from journal in portable text format.
- Verify Database... Perform integrity checks on database.
- Itemise File... List contents of internal format unload, subset or journal.
- Reload Database... Create database from internal format copy.
- Import Database... Create database from portable text copy.
- Delete Database... Delete a database from disk.
- Database Settings... View or modify database properties.
- Record Schema... View or modify record definitions.
- Secondary Indexes... View or modify record index definitions.
- Import Records... Create and populate record from ODBC or SIR/XS SQLServer.
- Write Schema... Write database definition as commands or report to file.

- List Stats Produces report of database statistics.

Tabfile

The tabfile menu allows you to manage the connection and creation of SIR/XS table files.

- Control Tabfiles... Controls connections to SIR/XS tabfiles.
- Connect Tabfile... Open an existing tabfile.
- Create Tabfile... Create a new tabfile.
- Verify Tabfile... Perform integrity checks on tabfile.
- SirSQL Invoke SirSQL.

Settings

The settings menu allows you to view and change various session settings.

- File Attributes... Shows files and internal names.
- Global Variables... Shows global variables and values.
- Preferences... Controls settings such as preferred editor, html browser and font sizes.
- Master Settings... Controls concurrent database access.
- Buffers... View, add or delete memory buffers.
- List Remarks Toggle listing of remarks.
- List Commands Toggle listing of commands.

Help

The help menu provides information on using SIR/XS.

- Contents Open the main contents page.
- Index Open the alphabetic index.
- Search... Lets you search for any word.
- What's New Display information about new features in this release.
- About SIR... Gives licence and version details.
- SIR on the Web Opens the SIR web page at www.sir.com.au

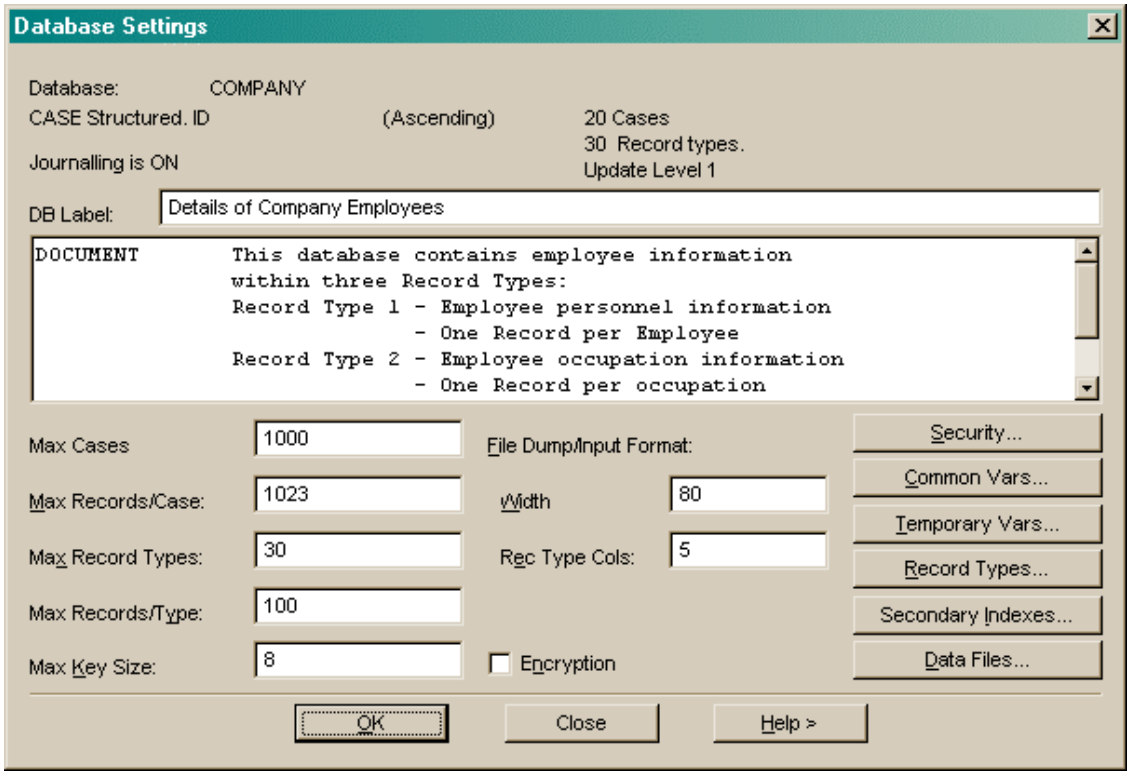
Database

The database is the primary means of storing data in SIR/XS. There can be any number of different databases each with multiple types of records. A single database can hold up to 4,095 different record types and there can be over 2,000,000,000 actual records in a database. Records can be grouped into Cases - a set of related records.

A record consists of up to 4,095 variables and each record can hold up to 32,000 characters of data. Each variable has a name and the database stores various information about the data. This can include such things as the format for dates or times, lists of allowable values and definitions of coding systems.

The description of a database and all the record types is known as the Database Schema. This can be created interactively through menus and modified as you extend your database design.

Schemas can be entered through screens or defined with schema definition commands.



Database Schema Settings

These result in an identical database, one is done interactively, the other is done by creating commands and running them in a similar way to a program.

You can modify the schema definition. SIR/XS allows alterations to the design even after the data is loaded without having to unload and reload data.

A SIR/XS session can connect to multiple databases and switch between databases making one database the default. Programs and utilities operate on the default database.

Whenever data is updated, it is maintained according to the dictionary definitions. Every SIR/XS component that accesses data from a database does so through the database manager which means that the data is always verified and conforms to the data dictionary.

Data is always referenced by name. Programs do not know anything about the physical organisation of data. Queries and applications are independent of the physical structure of data.

Utilities

Data can be loaded from existing files through a set of Batch Data Input utilities. There are also utilities to help manage a database. These provide statistics on the database, make backup copies of your data and allow you to move data to different computers.

Master

You can control concurrent updates to a database by multiple users by running a database server known as SirMaster.

Tabfiles

A second means of storing data is through tabfiles which hold relational tables.

Tabfiles can be created by SQL with the `SELECT` command and by VisualPQL with the `SAVE TABLE` procedure as well as directly through definition commands.

VisualPQL

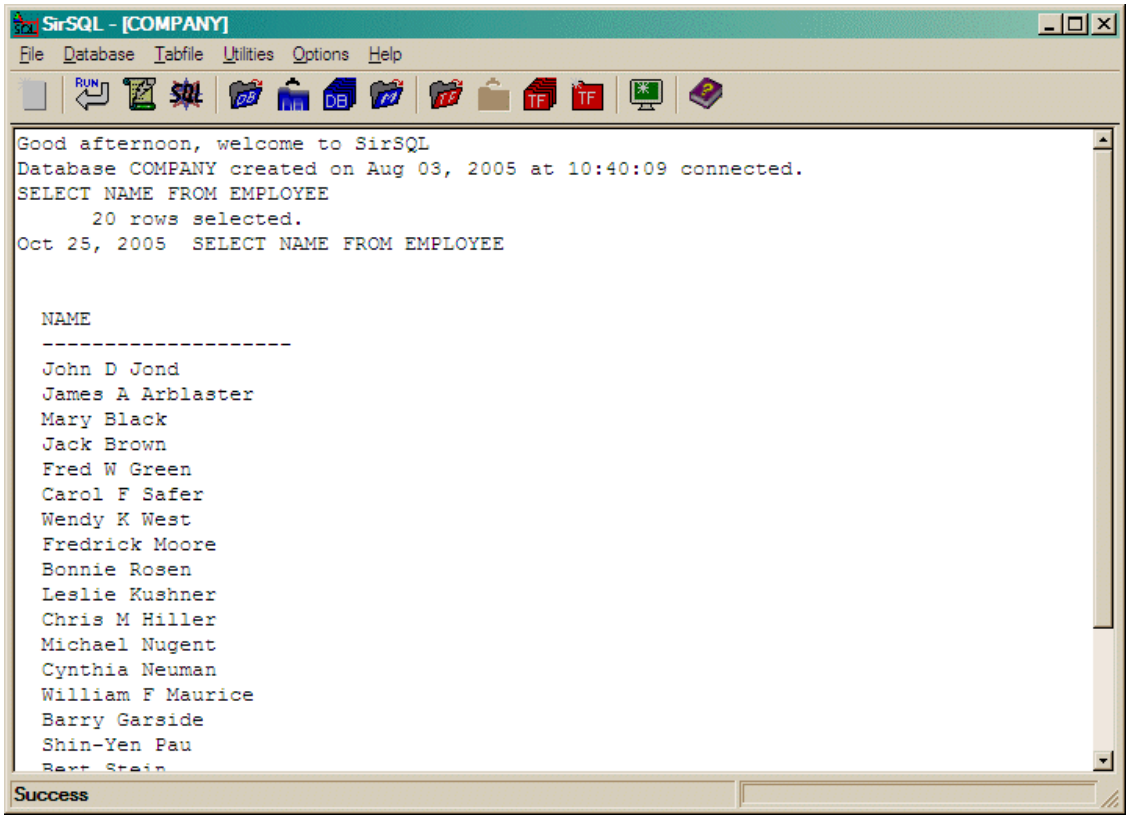
The heart of the application development system is the fourth generation application development language called VisualPQL (Procedural Query Language). This is a full programming language which includes a built-in set of data reporting and statistical procedures. You can use VisualPQL for simple tasks, such as producing a report, and for complex tasks, such as building full applications.

Programs and other routines are managed as part of the database known as the Procedure File. Developing programs requires the use of a text editor which can be any editor you want.

The procedure menu allows you to generate VisualPQL programs for a number of standard tasks without writing any code.

SirSQL

SQL (Structured Query Language) is part of SIR/XS and is an industry standard language with a menu interface or an English-like set of commands.



SirSQL

The primary use of SirSQL is to retrieve and display data interactively. SirSQL has many enhancements to simplify the interactive use of SirSQL and to take advantage of SIR/XS database structures.

PQLFORMS

PQLForms is a facility to generate a set of screens and program logic in VisualPQL that can then be used to enter, modify and view data interactively, one record at a time. A form is a set of linked screens, defined using commands or using a screen painter. These define what variables are on each screen, how they are displayed and edited, how a screen looks, and how screens are linked together. Once a form has been developed, it can be used by many people for data entry or for querying data.

Building an Application

A typical set of tasks to create an application is:

- Design a database and create the database in SIR/XS
- Load any data from existing serial files using Batch Data Input utilities
- Create and test report, analysis and enquiry programs using VisualPQL
- Create data entry screens using PQLForms
- Create dialogs and menus to link everything using VisualPQL

SQL is primarily an end user tool to access data interactively and typically does not require customisation.

COMPANY Database

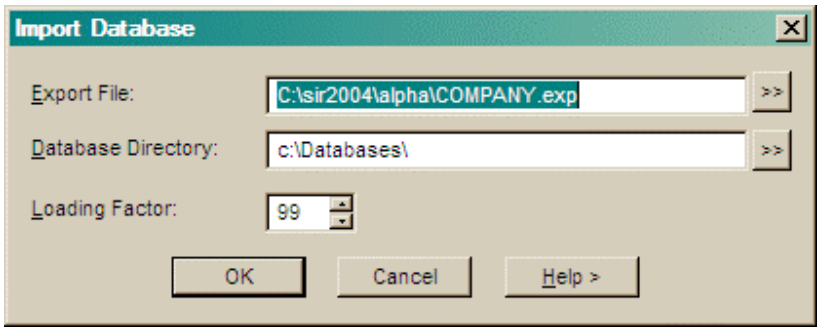
Start SIR/XS either by clicking on an icon or by entering the program name `Sir.exe`.

This starts the system which brings up the main menu.

There is a small sample database shipped with the software which also includes a number of example programs. This is the `COMPANY` database.

If you want to look at or run any of the examples and the `COMPANY` database has not already been created, you can easily recreate it as follows:

- To create a new `COMPANY` database in a directory of your choice, change to your directory and start SIR/XS.
- Select `Import Database` from the `Database/Recover` menu and select the `COMPANY.EXP` file and press `OK`. This rebuilds the database. The export file `COMPANY.EXP` is a text file that contains all the commands and data needed to define the database.



Company Import

When opening a database that already exists, you require the database name and any passwords that have been defined.

For the Company database, `COMPANY` is the database name, `COMPANY` is the database password, `HIGH` is read security, `HIGH` is write security.

This information can be supplied as parameters on the execution statement as
`SIR DB=COMPANY PW=COMPANY RS=HIGH WS=HIGH`
to avoid re-typing each time.

Having attached a database, a first task might be to retrieve some data and for this you may want to use VisualPQL.

VisualPQL

VisualPQL is a structured programming and application development language. It is a high level language with a large range of features including:

- Programs and Subroutines
- Variables and arrays
- Logical Block Structures
- Access to files
- Constructs for accessing data in a SIR/XS relational database
- Constructs for accessing data in tabfiles
- Building dialogs and menus for use in a graphical user environment
- Building applications to access data a record at a time in a graphical user environment
- A wide range (300+) of built in functions
- Built in Procedures which create output in various formats such as reports, cross-tabulations or output files for other software packages such as SPSS. A VisualPQL program can include one or more procedures, and can collect and select the required data from the database or from other sources and then use the procedures to produce sorted output, cross tabulations or other analysis.

VisualPQL is the major application development tool within SIR/XS and is extremely powerful. VisualPQL can be used for numerous tasks from simply analysing some data to building a complex, menu driven application.

A first program

The only way to learn a new language is to write programs in it. The first program in all languages is the same. Print the words: `Hello World`

In some languages, this is a big hurdle. However with SIR/XS, it is easy and straightforward. The VisualPQL program to display "Hello World" is:

```
PROGRAM
WRITE 'Hello World'
END PROGRAM
```

All you have to do is to create this text and keep it and you can do this either as a:

- Member - The database has a set of facilities for keeping and managing text, typically programs but sometimes instructions and other documentation. This is known as the *Procedure* file. When the database is exported or unloaded, procedures are kept with the database so this is a permanent place for text associated with a database.
- File - You can keep the source of your programs as standard operating system files which can be edited.

Assuming that you have the `COMPANY` database attached, you might put your test programs in a new family of procedures called, say, `TEST`

You use an editor to create the text. The system allows you to use any editor of your choice and also has a very basic internal editor. Use whatever program you would normally use to edit text and that you are comfortable using.

To check which editor is being used, go to the `Settings - Preferences` dialog. This has a check box for external editor and the name of your selected external editor. If the box is not checked, the name is greyed out and the system uses the internal editor. If you do not have access to an external text editor, use the SIR GUI editor. If you select an external editor, the system saves this from session to session. When using the external editor option ensure that the path and the name are correct otherwise trying to edit a program will not work.

Steps to run a first program

- Start `SIR/XS`. Attach the `COMPANY` database.
- Select the `Program` menu.
- Select `Members`.
- Possibly add a new family for your test programs.

- Select new and give a member name, say `PROG1`.
- Use the editor to type in your program. Exit the editor and save the program.
- Position to the new member and select 'Run'. The program is compiled and then executes. It displays the messages
-
- ```
  Start program compilation
  Start program execution
  Hello World
  End program execution
  ```

Congratulations. You have created and run that vital first program!

N.B. If you do not want to type this program in, it is supplied as member `EXAMPLE.HELLO`.
  ```

VisualPQL Syntax

See VisualPQL syntax for a full description. Briefly the rules for VisualPQL are:

- Each new command begins in the first text position of a line. For example, each line in the following program is a new command:
-
- ```
  RETRIEVAL
  PROCESS CASES
  PROCESS REC EMPLOYEE
  WRITE ID NAME
  BIRTHDAY SALARY
  END REC
  END CASE
  END RETRIEVAL
  ```
- Leave the first position blank to continue the previous command. Put a period (.) in position one to specify that this is a new command but that the command is indented for easier reading.
-
- ```
  RETRIEVAL
  . PROCESS CASES
  .  PROCESS REC EMPLOYEE
  .  WRITE ID NAME BIRTHDAY SALARY
  .  END REC
  . END CASE
  END RETRIEVAL
  ```
- Generally, commands and keywords cannot be abbreviated but some commands have synonyms. For example, `PROCESS REC` is a synonym for `PROCESS RECORD`.
- Comments can be put at the end of a line. Put a vertical bar (|) character after the command, before the comment.

  There is also a `COMMENT` command (synonym C or CC). This specifies that the whole line is a comment.

- Upper and lower case letters are treated identically for all commands, names, keywords, etc. The only places where upper and lower case makes a difference is in character strings (in quotes) or if using non-standard names (in curly brackets{}).
- A VisualPQL routine begins with a command such as `PROGRAM` and ends with a matching `END` command. The commands to begin a routine are:
  - `PROGRAM` - any main routine
  - `RETRIEVAL` - a main routine that accesses a database
  - `FORM` - a main routine that creates screens to do record at a time browsing
  - `SUBROUTINE` - an independent routine that is executed from other routines

Example VisualPQL programs

The mechanics of creating and running the programs are going to be taken as understood from now on. Before doing too much typing in, try saving and restoring data from members or files and you should be comfortable with the mechanics of using an editor.

The following program uses a formula to calculate Fahrenheit and Centigrade temperature equivalents. The program in its simplest form is:

```
PROGRAM
FOR FAHR = 0,300,20
COMPUTE CELSIUS = 5/9 * (FAHR - 32)
WRITE FAHR CELSIUS
END FOR
END PROGRAM
```

VisualPQL is a block structured language and the block used here is a `FOR` block. This assigns an initial value to a variable and then performs the commands in the block a number of times, incrementing the initial variable each time until the final value is reached. In this case, these values are expressed as constants with a start of 0, an end of 300 and an increment of 20. The end of the block is indicated by the `END FOR` command.

Because neither the `FAHR` nor `CELSIUS` variables were explicitly declared, they are implicitly declared. In the context they were used they had to be numeric.

Try running the temperature program (supplied as `EXAMPLES.TEMP1`) and look at the results. There is a default output format for all variables which may not be precisely what was wanted, but allows a very quick result to be produced. There are no comments in the program, nor any indentation to indicate which commands are in the loop and which are not. A slightly improved version of the program (`EXAMPLES.TEMP2`) might be:

```
PROGRAM
C** This program computes Fahrenheit and Celsius equivalents
C** between 0 and 300 fahrenheit
WRITE 'FAHRENHEIT CELSIUS'
FOR FAHR = 0,300,20
. COMPUTE CELSIUS = 5/9 * (FAHR - 32)
. WRITE FAHR(f6.2) 12T CELSIUS(f6.2)
END FOR
END PROGRAM
```

Note the use of lines starting with a period for indentation. All commands start in column one and periods are used to indent for readability. Indentation and the use of one command per line is recommended. The C in column one indicates a comment. C** is often used conventionally.

The `WRITE` can specify output formats (enclosed in parentheses after the variable) to align the output. The (f6.2) format indicates a floating point format 6 characters long with 2 decimal places. The 12T specification specifies that the next field starts in position 12. Two `WRITE` statements have been used, one to do a brief heading and one to output each line of results. A `WRITE` automatically writes a new line.

The following version (`EXAMPLES.TEMP3`) of the temperature conversion program uses variables to control the `FOR` loop and uses the `SET` command to initialise these. It uses the `SIMPLE REPORT` procedure and writes to a file rather than to the screen.

```
PROGRAM
C** This program computes fahrenheit and celsius equivalents
C** between a lower limit, and an upper limit in intervals
INTEGER*2 lower upper interval
SET lower,upper,interval(0,300,20)
FOR FAHR = lower,upper,interval
. CELSIUS = 5/9 * (FAHR - 32)
. PERFORM PROCS
END FOR
REPORT FILENAME = 'TEMP.LIS'
       PRINT  = FAHR CELSIUS
       NOTOTALS
END PROGRAM
```

More VisualPQL

Logical Conditions

Some commands are executed depending on particular logical conditions. Specify logical conditions in brackets. You can construct compound conditions using `AND, OR`, etc.

Because conditional tests are frequently required for very simple computations or actions, there is an `IF` command which is not block structured; it simply determines whether or not the command is executed. The action to be taken is expressed as a continuation of the `IF` command itself. For example:

```
RETRIEVAL
PROCESS CASES
. PROCESS RECORD EMPLOYEE
.  IF (GENDER EQ 1 ) WRITE NAME
. END PROCESS RECORD
END PROCESS CASES
```

Database Access

Begin a program that accesses the database with the `RETRIEVAL` command which opens the default database (if more than one database is connected, this is the last database connected or it can be set explicitly).

If the retrieval needs to create, modify or delete data on database records, specify the `UPDATE` option on the `RETRIEVAL` command. This opens the database for write access. Retrievals without the `UPDATE` option can get data from the database but cannot add, delete or modify data.

A very simple program which accesses the database might be as follows. This program lists the name of all employees:

```
RETRIEVAL
PROCESS CASES
. PROCESS RECORD EMPLOYEE
.  WRITE NAME
. END PROCESS RECORD
END PROCESS CASES
END RETRIEVAL
```

Things to note about this program:

- The initial command is `RETRIEVAL` (as opposed to `PROGRAM`) indicating that it uses the database.
- The `PROCESS RECORD / END PROCESS RECORD` is a block structure and the block of commands within the `PROCESS REC` block is executed for each `EMPLOYEE` record that is retrieved.
- `NAME` is a string variable on the `EMPLOYEE` record.
- The Company database is a *case structured* database. This means that all of the data records for a single employee are grouped together. One employee is one case. This is frequently a good way of organising data.
- The `PROCESS CASE / END PROCESS CASE` block processes every case. If you leave this block out, the program will still work but will give you a warning message.

Accessing Data from the Database

During execution, a retrieval accesses data with one of the Case or Record commands. These processing commands define a block of commands. Within a block, other commands may get values from or put values into the database. The processing commands are:

`PROCESS CASE` | `PROCESS RECORD`
: Retrieves a set of cases or records depending on specified key values.

`CASE IS` | `RECORD IS`
: Either retrieves or creates the specified case or record. The complete key must be specified.

  `NEW CASE IS | NEW RECORD IS`
  Is a variant which creates a new case or record if one does not already exist.

  `OLD CASE IS | OLD RECORD IS`
  Is a variant which retrieves an existing single case or record.

These blocks are terminated with the `END CASE/RECORD` command. On case structured databases, record blocks are nested within a case block.

The `PROCESS` commands are a looping structure and retrieve all matching occurrences of data. The "`IS`" commands access a single occurrence.

The record processing commands specify a record type and may specify a particular record or subset of records to retrieve. This is done by specifying values which are matched to the record keys or keys in a *secondary index*.

For example, the `OCCUP` record has a key of `POSITION`. To get the `OCCUP` record which corresponds to the current position value:

```
RETRIEVAL
```

```
PROCESS CASES ALL
OLD RECORD IS EMPLOYEE
. GET VARS currpos
. OLD REC IS OCCUP (currpos)
.   GET VARS ALL
.   WRITE ID NAME CURRPOS STARTSAL ...
. END RECORD IS
END RECORD IS
END RETRIEVAL
```

If there are no matching records, then the block of commands is skipped completely. In the previous example, the `WRITE` is not executed if there is no matching `OCCUP` record for an employee and thus that employee does not appear in the output. Selection criteria may be specified as constants or variables.

On a `PROCESS CASE` command there are options to select counts, samples or lists of cases. On a `PROCESS RECORD` command, there are a number of ways of specifying records to be selected. These include selecting records with keys from a specified value, to a specified value and in a given range. Records can have multiple key fields and the selection may be applied to a subset of keys. In a case structured database, the case id is implicitly used as the first key on record keys (not on secondary indexes). For example to select a sample of employees and find an average of their starting salaries in position grades 4 through 10:

```
RETRIEVAL
PROCESS CASES SAMPLE = .25
. PROCESS RECORD  OCCUP FROM (4) THRU (10)
.   COMPUTE AVGSAL = MEANR (STARTSAL);
            STDD   = STDEVR (STARTSAL)
. END PROCESS RECORD
END PROCESS CASE
WRITE 'Average Salary = ' AVGSAL '  Std. Dev = ' STDD
END RETRIEVAL
```

To retrieve all employees by name assuming a secondary index called `NAME_INDEX` has been defined on name:

```
RETRIEVAL
PROCESS RECORD EMPLOYEE INDEXED BY NAME_INDEX
WRITE NAME 25T ID
END PROCESS REC
END RETRIEVAL
```

Database

There are screens to define or modify the overall database information, the information about each record and the information about each variable within a record. These screens can be used to set up any kind of database structure or to modify existing database definitions.

As an alternative to using the interactive screens, there are a set of database definition commands. It is sometimes more convenient to create commands using a text editor and run these.

The database definition is referred to as the *schema*. There are options on the menus to view the schema and to write out copies of the schema to a file.

Creating a new database

To create a new database, select `New` from the Database - databases menu.

Specify the database name and, optionally, a password in the dialogue box.

As an exercise:

- Go to your directory, start SIR/XS
- Select `New` from the `Database/Databases` menu.
- Enter `TEST` as the database name and do not use a password.
- Click `Next`
- Congratulations - you have created a database. However, this database has no records defined and so cannot yet store any data. The screen now lets you enter some description of your database. Click `OK`, then `Done` to finish.

Example Database:

The following is a brief illustration of the type of data that might be held in a database.

```
DATABASE  SURGERY
CASE ID   PATNO
Record Type 1  PATIENT
```

PATNO	NAME	PHONE	ADDRESS	PCODE	DOB
1	James	425-1234	21 High St.	2056	15JAN65
2	Smith	364-9238	1A The Vale	3458	10FEB77

3	Jones	858-3289	32 Main Road	4754	30DEC54

`Record Type 2 VISIT`

PATNO	VISDATE	SYMPTOM	ACTION	OUTCOME
1	10JAN92	Headache	Aspirin	No change
1	2FEB92	Headache	Aspirin	Cured
1	12APRIL92	Footache	None	*
2	01JAN92	Joint Pains	Infra-red	*

`Record Type 3 GENERAL`

PATNO	HEIGHT	WEIGHT	INSURE
1	2.2	155	GA123/7
2	1.9	205	*
3	1.7	*	AE435-32

This example database is called `SURGERY`. It is a case structured database using the patient number (`PATNO`) as the case identifier. The data can be linked for any given patient. Data can be analysed by patient or by record type.

There are three record types: `PATIENT`, `VISIT` and `GENERAL`.

Each of these has variables of various types:

- `PATNO` is an integer
- `NAME, PHONE, ADDRESS, PCODE` and `INSURE` are strings of various lengths
- `DOB` and `VISDATE` are dates
- `HEIGHT` and `WEIGHT` are real numbers
- `SYMPTOM`, `ACTION` and `OUTCOME` are strings with a defined set of possible values or codes for these variables so it is possible to define these as `categorical`

All of these types of variables can be defined in the SIR/XS database.

Maintaining Databases

Even the smallest and shortest-lived database needs some maintenance work. As databases grow larger, become more complex and involve more people, their maintenance becomes more and more important.

There are several issues that affect all database users, particularly managers of databases that other people use.

Safeguarding Databases

Computers are subject to failure. Power failures, electrical storms and human error can happen. A database could be corrupted if the computer "crashes" at the wrong time. There are utilities to check whether a database is corrupt and utilities to attempt to recover from corruptions.

There are several things to do to protect against disaster.

Journaling

When journaling is turned on, the journal file keeps track of every change made to a database. Anytime that the schema or data is modified, a record of the change is written in the journal file.

If disaster strikes (e.g. disk crash, power failure) and a database is damaged, it can be completely reconstructed from backup copies and the journal file. When creating a database, journaling can be turned on or off. Journaling can also be turned on or off at any later time. If a database is easy to re-create from an existing computer file or is only going to be used for a very short time, turning journaling off saves time and disk space. Normally keep journaling turned on.

Backups

The `UNLOAD FILE` utility backs up a database by copying the database files to a single sequential file. The journal file can be used to recover a database by applying the changes it contains to an earlier version of the database. These files should be copied to external media and archived. How frequently a database should be backed up varies with the amount of update activity.

Always run a `verify` (from the `Database/Recover/Verify Database` menu) before taking a backup. This displays a single screen with some summary information and the message "Verification Complete. No errors were found." If any errors were found, use the `PATCH` option on `VERIFY` to attempt to correct the problem.

Writing Data

Use any of the following utilities (some from the `Database/Backup` menu and some from the `Data` menu).

- `EXPORT` creates a machine independent version of a database. This can be transferred and `IMPORT`ed to recreate the database and application on the new machine.
- `DUMP SIR FILE` creates a text file containing the data or a subset of the data formatted according to the specifications in the schema. This is a format suitable for input through the batch data utilities. This can be a way of passing data between systems or a way of saving data for re-input for one reason or another.
- `UNLOAD` backs up a database by copying the database files to a single sequential file. This can then be used to restore the database. How frequently a database should be backed up varies with the amount of update activity.
- `LIST SIR FILE` produces a readable or printable version of the data in the database.
- `ITEMIZE FILE` is used to check what is on a journal or an unload.
- `PURGE DATABASE` deletes the whole database.
- `DELETE RECORD SCHEMA` deletes a record definition.
- `LIST STATS` provides database statistics including important dates, database parameters, numbers of variables, keyfield size, number of schema redefinitions by record type and record counts by record type.
- `VERIFY DATABASE` checks the structure of a database.

Procedures

The procedures menu is a simple, intuitive way to produce reports and other outputs from a database which do not require custom programming. It generates VisualPQL which can be looked at, saved and used as the basis for other programs.

You can produce reports, cross-tabulations, interface files and various other outputs using the `VisualPQL Procedures`.

A VisualPQL program processes data and selects sets of variables which meet the selection criteria and uses a procedure to produce output.

To create a query, select the appropriate procedure, select the variables to output and any selection criteria to apply to the database records. Then run the query. Queries can be saved and re-run.

For most procedures, the only choices you must make are which procedure to run and which variables to use.