1986

ZIM: an evaluation of a 4GL product

David E.A. Wilson
University of Wollongong

The University of Wollongong

Department of Computing Science

Z I M

An Evaluation of a 4GL Product

by

David E.A. Wilson

Abstract

ZIM is a database product based on the Entity-Relationship database model. It was evaluated on a Sperry 5000/40 running UNIX System 5.2.

Z I M

An Evaluation of a 4GL Product

by

David E.A. Wilson
Department of Computing Science

1. Introduction

ZIM is a database product for PC-DOS, MS-DOS, UNIX & XENIX based microcomputers. It is based on the Entity-Relationship database model with an integrated data dictionary. In addition, it has a "fourth generation language" interface between the user (or programmer) and the database. This language includes a toolbox of predefined functions for manipulating text, numbers, dates and sets of numbers.

2. Product Identification

2.1 Name

ZIM Version 2.5 (Sep 85) for UNIX and XENIX Operating Systems.

2.2 Author

Zanthe Information Inc, Ottawa.

2.3 Hardware Environment

The evaluation of ZIM took place on a 68010 based Sperry 5000/40 running UNIX System 5.2. Two different terminals were used - a Volker Craig VC4404 and a Sperry SVT-1220.

3. Installation

The documentation provided with ZIM gave full instructions for installation on Spectrix Microsystem, IBM PC/AT or NCR Tower 1632 computers but did not mention the Sperry 5000/40. As ZIM was supplied on a "tar" format tape the instructions for the NCR Tower could be used (by changing the device from the floppy disc to the streaming tape drive).

The next requirement was for the termcap entries to be modified (or created). The problems found with this procedure were as follows:

(a) The Volker Craig VC4404 used in the evaluation has function keys returning single control characters. As one of them was a control-\ (an ASCII value of 28) this caused a quit signal to be sent to ZIM. No mention was made in the manual about the need to change the interrupt & quit characters if they conflict with function key assignments.

(b) The Sperry SVT-1220 does not have a backspace key (control-H or an ASCII value of 8) forcing the delete key (ASCII 127) to be used in its place as the erase character. While UNIX allows this substitution, ZIM always uses control-H for erasing the previous character when in forms mode, even if the termcap definition includes the capability "kb=\177" indicating that the backspace key on this terminal sends a delete character. On the VC4404 (and many other terminals) the left arrow key is identical to the backspace key (both sending a control-H). In this situation, both the backspace and left arrow keys are taken by ZIM to mean "move the cursor left one space" and the erase previous character function is unavailable.

(c) ZIM offers no support for the 132 column mode of the SVT-1220 and other terminals.

Provided the instructions for loading ZIM are modified to match the distribution media supplied for the Sperry, the average user would have no trouble at all installing ZIM, and only slight problems customising the termcap entry.

4. Disk and Memory Requirements

When first loaded, ZIM requires the following amount of space for disk storage:

- Tutorial - 336 Kbytes
- Example Database - 264 Kbytes
- ZIM & IDF - 1175 Kbytes

The IDF is an Interactive Definition Facility which allows the unskilled user to very quickly get an application prototyped with forms and input/update procedures.

The relatively small size of ZIM (in comparison to many other products) allows its use on small UNIX systems with only 20 to 30 Megabytes of disk storage available.

5. Tutorial and Supplied Example Database

The tutorial provided with ZIM is an excellent introduction to the basic concepts required to build and utilise a database using ZIM. The tutorial starts by explaining what an entity set is (ZIM's term for a table), how it is created and how it can be combined with other entity sets through relationships. As the user progresses through the lessons, ZIM commands are introduced when appropriate and with examples to show their use and variations.

Two lessons are dedicated to forms. In ZIM, forms play an important part in the use of a database as they provide a user friendly interface between the entity sets and the user. Forms also allow automatic validation of data input to prohibit the entry of invalid data into the database.

One lesson is dedicated to report generation and shows the user how to generate simple reports (it would take a user some time to learn to use all of the many report commands).

The example database gives a simple model of the employees, departments & projects in a small company. Prewritten reports are included and the source files are provided so that the user may learn from the example.

6. Relational Completeness

ZIM is based on the Entity-Relationship Model of data, claimed to be an enhancement of the relational model. However, ZIM fails to meet all the requirements of a relational system as defined by Ted Codd[1]. A number of these failures are listed below.

(a) Views - ZIM has no concept of a view. It is not possible to define a view that appears to the user as an entity set but which is in fact two entity sets joined by a relationship. Nor is it possible to define a view of an entity set with fewer columns than the original - to hide salary details from normal users for example.

(b) Validation - ZIM cannot store validation criteria in its data dictionary. This forces the user to make explicit tests in procedural code or to use forms mode for all data entry (ZIM forms do have data validation).

(c) Keys - ZIM has no concept of Primary and Foreign keys. This results in a number of problems, from an increase in procedural code to keep the data base referentially intact (ie. Adding a new employee to a non-existent department), to allowing duplicate records and records with null keys.

ZIM's unique key selection feature is often insufficient as in a number of situations there may exist no single field which uniquely identifies a record. However, by combining a number of fields together one obtains a key with the required uniqueness (ie. a bank account number is only unique when prefixed by the branch number of the bank at which the account resides).

(d) ZIM's data dictionary is not active - it does not represent the current definitions if changes have been made since the last CREATE command - and there does not seem to be any way of asking ZIM what the definitions it is currently using look like.

On the other hand, the entity relational model does have a number of features not present in the purely relational model. They include:

---

1 In the article "Is your DBMS really relational?" in Computerworld Australia February 7, 1986.

(a) Fields defined within relationships - Using this feature it is possible to store data in a relationship (ie. in a database of people a relationship might be "marriage" and the number of children by the marriage should be stored once in the relationship rather than in one (or both) of the parents records - this is also useful in the case of multiple marriages etc).

7. Forms

ZIM's form design capabilities are quite good, allowing the designer complete flexibility in the layout of the form. Individual fields may be moved about the screen and highlighting may be used on both prompt and data fields. Validation criteria may be specified for each data field and an error message will appear at the bottom of the screen if the user enters invalid data. Forms may take advantage of certain terminal attributes (if the terminal in use supports them) such as inverse video, bolding, underlining & blinking characters.

When the form is used, the function keys (plus Enter and Escape) may be set to terminate or exit from the form. When the form returns to the user program it is able to test a number of predefined variables to find out what key caused the form to end, what field the cursor was in and even which direction the cursor was moving (used for fields which transmit as soon as the cursor leaves them). This allows user written validation (if ZIM's validation is insufficient) and single key selection of menu items (as used in the tutorial) and many other possibilities.

8. Reports

The REPORT command in ZIM offers a great number of options to the report designer, allowing full control over field placement, headings and control breaks. The masking capabilities are quite impressive, giving COBOL style masks for numeric items (floating $ signs, cheque protection with asterisks and CREDIT/DEBIT notation) while for character fields it exceeds COBOL in allowing insertion editing where successive '?' characters are replaced with characters from the data field. Reports may be generated from the terminal but it is more likely that a report of any complexity would be stored in a user command file and invoked by name.

9. Procedural Code

ZIM allows the use of procedural code where the non-procedural commands provided are insufficient. Strangely, the WHILE command may not be used to create a loop from the terminal as it only works from within a user command file. Furthermore, the following piece of code is the recommended way of processing a group of records in a record by record manner:

```
find <entity set> where <condition> -> tempset
while $SetCount > 0
    <process record>
    let $SetCount = $SetCount - 1
    next tempset
endwhile
```

The need to explicitly decrement a counter (a 3GL construct) appears out of place in a system which claims to "offer significant advantages over these so called 'fourth generation' systems". Looking through the examples provided with the system it appears that a great deal of procedural code is required to overcome the limitations of the database (key validation etc).

10. Interactive Definition Facility

The definition of entities and associated forms can be a difficult task when first attempted. The IDF offers a menu driven approach to the task. Using the IDF, generation of a database is reduced to the answering of questions displayed on the terminal.

Two methods are offered - creating individual definitions or using the Automatic Application Generator. Using the AAG the entity set and its fields are defined, a default form is created and procedures to add and update the entity set (using the default form) are written.

Using this method, the user is taken through the problem step by step without the need to learn any commands. At the completion of the database definition the user has code fragments for updating the database which may be expanded to form a complete application. The simple form can be used as the basis for a more complex customised form by highlighting fields, adding headings, prompts or validation criteria.

11. Manual

The manual provided with ZIM 2.5 is a vast improvement on the previous version. It has a good layout, is well indexed and is divided into four main parts - setting ZIM up, the tutorial, basic concepts (which describes the various parts of ZIM in a brief overview) & the reference section (which describes each command and function in full detail). In addition there is the appendix which describes the utility programs supplied with ZIM (for creating an empty database or fixing damaged database files) and the index for the entire volume.

12. Errors

During the evaluation of ZIM, a number of errors were found in the Tutorial, Example Database and most importantly, in ZIM itself.

11.1 Tutorial Errors

(a) Lesson 2, page 10 - When the newly created entity set is listed, it is found to contain one record, rather than being empty as stated.

(b) Lesson 3, page 18 - If the command "LIST ALL FORMAT Invoices.Invnum Reference.Invnum" were executed as shown, the output would have only two columns labeled InvNum, not the four as shown on this screen (2 x InvNum and 2 x PartNum).

(c) Lesson 3, page 27 - When the command "list all currentset format partnum invoices.invnum" is executed, an error message is displayed instead of the expected table. The error message reads "*** Error *** The field 'partnum' is known within more than one component of the set. Use name qualification".

(d) Lesson 4, page 17 - It is stated that the data types of document fields must be character, but the example given has a field called balance with a data type of numeric.

(e) Lesson 5, page 1 - This screen states that Forms and Displays will be discussed in lessons 5 and 6. It should read lessons 6 and 7.

(f) Lesson 5, page 13 - NextInvNum is stated to have a data type of char but when listed is shown to be numeric.

(g) Lesson 6, page 2 - a minor mistake - the sentence "Let's take a look a the fParts form" should read "Let's take a look at the fParts form".

(h) Lesson 6, page 10 - the reference to lesson 6 should read lesson 7.

(i) Lesson 7, page 11 - the user command updatecusts has an error such that changes to the balance field of one customer are carried through to that of the next customer displayed.

(j) Lesson 8, page 1 - the reference to lessons 5 and 6 should again refer to lessons 6 and 7.

(k) Lesson 8, pages 11 & 17 - the examples given are one line too long to fit on a standard 24 line screen.

(l) Lesson 8, summary - another minor error - the last occurrence of the word ZIM in the summary has an embedded \e character.

(m) Lesson 9, pages 3 and 6 - The field "funds" (added on page 2) does not appear in the fields listing.

(n) Lesson 10 - this lesson is missing completely.

11.2 Example Database Errors

(a) If a syntacticly incorrect selection expression is entered, the error message "*** Error *** -> is not allowed at this point in the command" appears on the bottom line of the screen. This causes the screen to scroll up one line and the prompt fields then no longer line up with the data entry fields and the screen looks a mess. The only cure for the user is to exit from the screen and re-enter it.

11.3 ZIM Errors

(a) If the name of an entity set is mispelt, the error message reads "There is no current set or the name is unknown" with two spaces between "name" and "is". This is where the incorrect name should have been listed.

13. UNIX Specific Features

When run under UNIX, ZIM has a number of extra features that are not present in the DOS version. They include:

(a) With the release of ZIM 2.5 the ability to read (write) from (to) pipelines has been added. This solves the problem of writing to the line-printer on a multi-user system. With the previous release of ZIM, the document "printer" could refer to the actual device such as "/dev/lp" but on most UNIX systems this would not be writable by normal users. Instead a spooling program would be called and it would queue the various users files and print them sequentially. Using the document pipe feature it is now possible to define the document "printer" as "|lp -s" which under UNIX 5.2 would cause the output to be spooled to the printer without any messages appearing on the users terminal.

(b) The user may now select his favourite editor for use within ZIM by assigning its name to the variable "editor".

(c) Interrupt Handling - ZIM allows a user program to disable the standard UNIX interrupt key.

In its default state, when interrupted ZIM will ask the user if he wishes to continue in ZIM. If the answer is yes, ZIM will continue to execute the current command until the current change is complete and returns you to ZIM's command level. If the answer is no, ZIM will close all its files and stop immediately (possibly leaving the database in a corrupt state).

When a user command has been fully debugged, the "breakable" flag may be turned off to avoid users accidentally corrupting the database. If this command is used it is still possible to stop a runaway ZIM user command by using the UNIX kill command from another terminal.

14. Multi-User ZIM

When ZIM is run under UNIX or XENIX, it comes in two flavours - single-user and multi-user. In single-user mode the data dictionary may be updated while under multi-user it may not. This prevents one user from affecting another.

When running under multi-user ZIM, a user program may group together a number of ZIM commands to form a single transaction such that if any one command in the transaction fails, no changes at all will be made to the database.

ZIM uses the operating system file locking to lock either small groups of records or an entire file while it is being updated. This ensures that the other users see a consistent database during updates.

15. Overall Impressions

While the entity-relationship model does offer a number of advantages over the relational model, in this implementation a few features have been lost. They include:

(a) The ability to perform cross or join operations on domains in ways that were not thought of before without the necessity of adding the join to the data dictionary. In single user ZIM this does not present an insurmountable problem as a new relationship may be defined in a few minutes. In multi-user ZIM however, the users cannot alter the data dictionary and so cannot add new relationships.

(b) The ability to define views which either limit some users to a subset of the available fields in a domain (for security reasons i.e. salary details) or join a number of domains into a single effective domain in a way that is transparent to the user (by allowing both enquiries & updates).

Inspite of the problems and limitations noted in this report, ZIM is a well documented and easily used product. Using the tools provided with ZIM, most end users would have no difficulty in creating simple applications with display screens and printed reports.