Storing Hierarchies in the ERROS Database

In ***Horizontal or Vertical? Beyond the Relational Database,*** I described in outline the structure of the ERROS database, which stores each attribute iteration as a separate record, linked to other records with a multipart key field.

I explained how the structure of the database is defined in the ERROS database – in other words the ERROS database is defined by itself, rather than in program code. As each entity type is given a number which is part of the key field, the data and application definitions and user data could all be stored in a single file, but, for operational reasons, about ten files are used. When a new entity type is created, no new file is required.

KEY FIELD							CONTROL DATA	USER DATA		
Entity Type	Recd Type	Record Number	Attribute Number	Recd Sequ	Compressed Record Id.	Record Id. Suffix	Dormant Marker	Record Data	Record Number	Link Field
12345	0	000000	00000	0	Smith, Fred (compressed)	001		Fred Smith	65737	
12345	0	000000	00000	0	Smith, Fred (compressed)	002		Fred Smith	76543	
12345	0	000000	00000	0	XYZ Company (compressed)	001		XYZ Company	57692	
User selects XYZ Company and is then presented with a list of attributes available in application 32 about the XYZ Company from which he can select (or request all). Although very similar in structure to user data, the menus are stored in a separate file from the user data.										
12345	1	000032	00000	0	address (compressed)	001		address	1	
12345	1	000032	00000	0	credit limit (compressed)	001		Credit limit	3	
12345	1	000032	00000	0	customer type (compressed)	001		customer type	4	
12345	1	000032	00000	0	telephone number (compressed)	001		telephone number	2	
If he selected address, the database handler would retrieve the following record. The record type ia now 0 as this is a data rather than a menu record.										
12345	0	057692	00001	0	23 Acacia Avenue (compressed)	001		23 Acacia Avenue	1	
The user required a facility to be able to store directions to some addresses. If he selects the address, ERROS will then retrieve a menu about the XYZ Company's address. It could contain many records..An identical menu record might be in another application that accesses customer addresses. Because this is a menu record, the key field stores the number of the application – 32 – and the record type is now 1 . The attribute number now has the number of the attribute "address" which is 1. This is not the number of the address record 23 Acacia Avenue which happens to be the same. Because the ERROS database stores all data in its context, ERROS does not get the numbers confused.										
12345	1	000032	00001	1	delivery instructions (compressed)	001		delivery instructions	5	
The record type is now 0 as this is a data rather than a menu record. The record number is 1, a unique number generated by ERROS when the XYZ company was related to the address. If the XYZ company had another address or another company was based at the same address, other unique numbers would be created. This is a text record and multiple records can be used if required.										
12345	0	000001	00005	4	00001			Turn left at the pub		

This means that the database handler can access an entity type by name, in this case customer name, then a customer by name, XYZ Company, then an attribute of customer by name, address, then an address by "name", 23 Acacia Avenue, then an attribute of address by name, delivery instructions, and then text representing those instructions.

Thus, in a hierarchical sense, we have gone down 6 levels to get to the final text.

.entity type/customer/XYZ Company/address/23 Acacia Avenue/delivery instructions/text

ERROS allows the "name" at each level to be up to 64 characters long, so that means that we would need to be able to concatenate these together and that might mean a search term of 384 characters, yet, in ERROS the total key field length is only 28 bytes long. This is part of the unique patented concepts of ERROS which allow a depth of 999,999,998 levels in a hierarchy, all within the 28 bytes. In many cases the number of levels might be very much larger than the sample above and the search term much longer.

In other database systems, even going down 6 levels would be a problem and might mean that the length of the index or key field would need to increase by 64 bytes at each level.

The ERROS database can also store bi-directional relationships and the user can, subject to his or her security level, navigate the relationships in either direction at the same, very high speed. The ERROS mechanism for storing relationships will be described separately.

Because all ERROS data is stored in very few files, and these are opened when the operator signs on, no joins are required and no files have to be opened when the operator elects to navigate a relationship. Relationships can also be hierarchical – in other words a relationship can be stored with another relationship.

KEY FIELD							CONTROL DATA	USER DATA		
Entity Type	Recd Type	Record Number	Attribute Number	Recd Sequ	Compressed Record Id.	Record Id. Suffix	Dormant Marker	Record Data	Record Number	Link Field
12345	0	000000	00000	0	XYZ Company (compressed)	001		XYZ Company	57692	
User selects XYZ Company and is then presented with a list of attributes available in application 32 about the XYZ Company from which he can select (or request all). Although very similar in structure to user data, the menus are stored in a separate file from the user data. Menu records define security for the attribute, presentation layout, etc.										
12345	1	000032	00000	0	address (compressed)	001		address	1	
12345	1	000032	00000	0	credit limit (compressed)	001		Credit limit	3	
12345	1	000032	00000	0	customer type (compressed)	001		customer type	4	
12345	1	000032	00000	0	telephone number (compressed)	001		telephone number	2	
12345	1	000032	00000	0	orders received (compressed)	001		orders received	6	
A new attribute, orders received, has now been added. This has automatically been allocated the number 6 by the database handler. These records are stored in ascending date and time sequence (record sequence 2). If the user selects this attribute, it instructs the database handler to read the records backwards and display the orders received with the most recent one first. If there are too many orders to fit on the screen, the user can page back through the orders, or, if he knows the date of the order, he can type that date in full or part (e.g.year and month without the day of the month). If that doesn't retrieve the correct order, the user can page backwards or forwards from that date without entering it again.										
12345	0	057692	00006	2	20161101104259	001		Sales order GBP 907.65	3015	
12345	0	057692	00006	2	20170603100500	001		Sales order GBP 436.98	4287	
12345	0	057692	00006	2	20170603145242	001		Sales order GBP 2724.00	4321	
The record number here is the order number. These records are bi-directional relationships with a record in the Customer entity type and another record in the Sales Order entity type, the two being connected by the link fields which the operator does not see. If the user clicks on a record, the database handler will take him or her straight to the relevant order in the Sales Order entity type (entity type number 14367). As the records for the different entity types are all stored in the same file, navigating from one entity type to another does not require a join, no files have to be opened or closed and no new programs are required to read the records of the new entity type. Amongst the records for the order will be the products ordered (attribute number 7)										
14367	0	004321	00007	0	board room chair (compressed)	001		Board Room Chair 20	6742	
14367	0	004321	00007	0	board room table (compressed)	001		Board Room Table 1	7580	
14367	0	004321	00007	0	filing cabinet (compressed)	001		Filing cabinet 2	321	
The record number here is the product number and the details such as quantity, price, total cost, VAT would be displayed on each line. The records here are stored in alphabetic sequence of product name so, even on an order with hundreds or even thousands of lines, any line can be retrieved immediately by product name (or generic name, e.g. "board").. The records could also be accessed immediately within the order by product number. These records are also relationships, in this case with the products. The operator can click on a product and, subject to his security, can navigate to the record for that product and see other orders, whether there is stock at another location, when next deliveries are due, etc.										

Clearly there need to be many record formats to store, for instance, personal names, sales order header records, sales order detail lines, product records, product movement records, etc. Apart from menu records and one or two other controlling records, such as operator profiles, the ERROS database handler does not need to understand the layout of any of the record formats that might be found in any standard commercial or humanities application. The record formats are not identified by the attribute name or number but by a simple method which will be described separately.

22nd June 2017

Rob Dixon

rob.dixon@erros.co.uk