ERROS – Multiple Entity Types, Groups and Sets, Ontologies and Taxonomies

The XYZ Company may be a customer whose data would logically be stored in the Customer entity type, but initially they may have been a prospect and perhaps their data then would have been in a Prospect entity type. As a customer, they may act as a retailer or as a wholesaler, or as an agent. Each of these entity types will have common attributes, such as address and telephone number, but there will be others that may only be relevant to one of the entity types. The XYZ Company might also be a Supplier and this would have its own attributes.

But the company has only one name, and it may have only one address and one telephone number, and it should not be necessary to store these in each of the entity types to which it applies. In any event, if someone from the XYZ Company telephones and the call is taken by an employee who does not know of them, it will take them too long to look in each entity type to be able to respond quickly to the caller. Perhaps the employee heard the caller's name but was unsure of the company name. In ERROS applications, these problems can be overcome by using another entity type – Contacts – and putting all people and companies names in that so that the person answering the phone can look for the caller's or the company's name. He does not need to know whether the company is a customer.

All contact records could have an attribute "is". This would be a relationship with an entity type "Type of Contact" and this would contain other entity types, such as "individual" and "business". When a record in one entity type, in this case, a Contact, is related to another entity type, it immediately also becomes part of that entity type as well as ERROS allows entities to belong to multiple entity types. The entity inherits the attributes of the entity type under which it is being considered.

Although ERROS allows any entity can belong to multiple entity types, the operator can only create relationships defined by the application developer in the ERROS database.

The caller **is** an individual and individuals can have an attribute "employer". This will store his employer's name as a relationship and the ERROS application will display that and perhaps the company's address and telephone number and the employee's email address at his employers and maybe his job title and department. Clicking on the employer's name will lead to further details about the business.

The XYZ Company **is** a business. Although individuals and businesses both have addresses and telephone numbers, businesses have employees rather than employers and they could have an attribute "relationship with us". This would define whether the XYZ company was a prospect, or a customer or a supplier, etc. These are also entity types and, if the operator clicked on customer, he or she would be able to look up sales ledger or sales order enquiries or create a new sales order, or for supplier would be able to look up purchases ledger or purchase order enquiries or check stock levels of products supplied by the company.

If the caller's telephone number is visible with caller display, then the caller could go to the entity type "telephone number", type in the number and ERROS would immediately display the name of the contact, and depending on whether it is an individual or business, will display the relevant data about the caller. Alternatively, having asked the caller for the zip or post code, the operator could access the contact data via the "address" entity type. For a customer querying an order, the operator could go to sales orders and type in the order number if the customer has it to hand or perhaps find it by date.

It is worth emphasising that all of the above facilities and many more can be created incrementally, without an upfront detailed system specification, without physical database design and without any programming and that the application will have extremely rapid response times that won't noticeably deteriorate as data volumes grow.

There could be a diary for all contacts, both individuals and businesses, and the operator could look in this so see a list of events in date sequence. If he recognises that the caller's query relates to one of the entries, he can click on it and immediately navigate to the event.

In the example above, we may consider “Contacts” as a primary entity type and “Individual”, “Business”, “Customer”, “Supplier”, etc. as secondary entity types. There are menus that relate to each entity type, so that when an operator accesses a contact record, ERROS will retrieve a contact menu as this is a primary entity type. However, when an operator accesses an entity in a secondary entity type, the application can be set up so that ERROS accesses either the menu for the primary or the secondary entity type, although the secondary menu would be more usual.

Data belonging to entities in a primary entity type will be stored in that entity type, but data belonging to an entity in a secondary entity type can be stored in either the primary or the secondary entity type and this will be defined by menu records. If it is stored in the primary entity type, it can be accessed from the menu of any secondary entity type to which it belongs. If the data is stored in the secondary entity type, then it can only be accessed when the entity is being considered under that entity type, whereas, if it is stored in the primary entity type, it can be accessed under any entity type. The menus will determine just which attributes are accessible in any application. There may be occasions when it is appropriate to store data in the secondary entity type but these are explained elsewhere.

Because entities can belong to multiple entity types, they can belong to a group or set entity type. This can be very valuable in a commercial environment. It is vital in the humanities for storing ontologies or taxonomies. The unique structure of the ERROS Connectionist database allows any entity in any entity type to be classified as belonging to any entity in any entity type and that in turn can belong to any entity in the same or any other entity type. Any “child” can belong to multiple “parents” and each of those can have multiple parents. There is no practical limit to the number of levels. Users can travel up and down all hierarchies without limits, and exit or enter at any level. They can find out in which hierarchies a record is contained without having to search any of the hierarchies, with obvious performance benefits. ERROS also supports synonyms which allow any term, name, etc. in any entity type to be found by using multiple alternative identifiers.

Rob Dixon

4th July 2017

rob.dixon@erros.co.uk
www.erros.co.uk