Structuring Your Database - Orixa Standards and "How To" Guide

Orixa uses a set of standards to manage the creation of your application (App).
When your App opens it analyzes the contents of your database and system-tables. The structures it sees in the database and the contents of the system-tables will determine the features it builds into your App.

This document summarizes the Orixa framework standards.

The Orixa framework accesses data from data-tables in an ElevateDB database. The framework automates the construction of many complex elements of the user interface, so that users can easily add, edit and view data. Orixa programmers structure their Apps following framework standards and then write SQL scripts which are stored in the "System tables" (these have the names "BusinessObjects", "Reports", "Searches", "Resources", "Types" and "Status"). These SQL scripts control the App and ensure it shows the desired data.

The full set of rules for creating and decorating an Orixa BusinessObject data-table is quite long. The easiest way to understand it is to look at the definitions of data-tables in an existing system and review how the definition of each table effects how it works. The "Create New BusinessObject" tool automates the options to create almost all the standard Orixa features, so it is not necessary to know them in detail prior to starting to develop and extend your own system.

Framework elements: Database

Data-table requirements (these rules must be followed for all tables)

- All tables must have a ID column as a primary key, and the value of this column must be set either via the "UID" univeral ID function, or by setting the ID of the table to match the ID of a "Master" data record that controls the rows in the table.
- All tables must have a "DateCreated" Timestamp column to identify when the record was created. Users opening records will not be able to edit this value. It will be set automatically by the database.
- Wherever possible try to give a table a column called "name" or "fullname". This can either be a simple VARCHAR, or it can be a computed column which brings together values from a number of columns in the table. The "name" column is not essential, but it is useful as Orixa has many systems to return lists of records, and these are much easier to generate if the table has a "name" column. If no "name" column exists, a more complex SQL script will need to be written to return a "name" using system-tables.
- Foreign-Key constraints are used to control the relational links between data-tables. Initially this is hard to understand, if you are new to database-programming, or used to using Excel spreadsheets to manage data in a "flat file". All data-tables in Orixa are linked together into a wider relational-map. This is always done using a link from the "child" table to the "parent" table on the ID Column. Standard writing style is for the "child" table column to be given the name "<parenttable>ID". (Example: A child "SalesItems" data-table would contain a column called "SalesID" and a CONSTRAINT "SalesID" which would link to the "Sales" data table on the ID Column).

Data-table standard columns (these rules are optional)

There are many commonalities in the types of data that are commonly stored in databases. Orixa works hard to try to standardize and normalize the names used by tables, so that the developer does not have to remember the name used for a column in a particular table.

Developers can use any name they like for many columns in the database, but if they follow Orixa standards it will make maintenance and support of the database much easier in the long-run.

- If an App requires a "Counter" for a table so you know "this record is record number 1, 234", then a field should be added called "OID" (=Orixa ID) with the following definition:
- `"OID" INTEGER DEFAULT OID('[TableName]')`
- The above definition will create a unique Counter for the table "TableName", so as records are added the OID will increment. This field will be separate from the table's "ID" field which is a counter **for the whole database.**
- If a table contains records which are regularly "completed" (example: A table of "Sales" data), then add a Boolean (true / false) column with the name "Complete".
- If a table contains records which may be "current" for a period of time, then cease to be used (example: A table of "Emails" data) then add a Boolean (true / false) column with the name "Current".

- Date columns: The column-name "DateDone" is the "go-to" name for a date-column wherever possible. If a table holds data relating to periods of time (example: a table of "Holidays" data) then two date columns, with the names "DateStart" and "DateEnd" should be used.
- The framework also has a set of built in behaviours relating to any columns with the name "DateEdited". If a data-table includes a column with this name, the user will not be able to edit it, and whenever the record is updated, the value of this column will be updated to match the current date-time.
- Number columns: Orixa has standard behaviour built in for tables with columns called "Value" or "Quantity" (example: a table holding rows relating to Purchases of goods).
- The framework also has default behaviour for columns with names: "AuthorID" this will automatically be populated with the name of the person who created the record, and updated with the name of anyone who edits a record. "StaffID" this will automatically show a list of "staff" in a business (these being records saved in a "Staff" data-table).
- StatusID: The framework has a system-table, "Status" which contains a "LogicalOrder" column and a "Name" column, as well as a "LinkTable" column. This allows the creation of a series of "status" values for a datatable (example: a "Purchases" table might include status's "1. New Order", "2. Confirmed Order", "3. Warehouse Processing Order" etc.). When you are creating a table that contains rows which change status over time, add a "StatusID" column. Adding a "StatusID" column to the database, with a suitable CONSTRAINT automates the creation of the systems in the database to hold this type of data list structure. The Status system-table also includes a **color** column which is used to hold a colour-value. This colour is pulled through and used in parts of the App user-interface.
- TypeID: The framwork has a system-table "Types" which contains "Name", "LinkTable" and "LinkField" columns. This allows the creation of a "lookup list" for a table. Add a column with a name "SomethingTypeID" (example "ProductsTypeID" for a "Products" table or "CustomersTypeID" for a "Customers" table). The Types system-table also includes a **color** field.
- AuthorID: If an Integer Column with this name is added to a table, whenever a table is updated the ID of the logged-on user will be placed in this field. The AuthorID therefore shows the "last user who edited" a record.
- PercentUsed: If a Computed Float Column with this name is added to a table, when displayed the data will show as a "progress bar" with value displayed as a green-strip between 0 and 100. Note that this is a read-only property.
- Orixa uses the standard that any field in a table that can have a simple, repetitive name should be given one. For example most "long text" fields are given the field-name "Description" or "Memo". This simplication and generalization makes it far easier to write SQL and reporting code that works well.

Note that Orixa contains tools which will automatically create BusinessObjects which follow Orixa rules, so developers do not need to learn them all exhaustively.

Data-table "Description" automation

The Orixa database includes the capability to "decorate" the definitions of the database elements such as tables, table-colunns, functions, views and procedures etc., with Descriptions. You App reviews these decorations at start-up to help build the rules for your data.

Details of decoration are moderately complicated, and not terribly easy to remember. However, it is always possible to look at the database and review decoration. Your own system is a useful reference-point for starting any new system extension.

SQL Statement:

```
SELECT Name, TableName, Description
FROM Information.TableColumns
WHERE Description IS NOT NULL
```

If the above statement is run, a grid will open which lists all the Description data which has been used to decorate columns in your system. You can use any of these descriptions on other data-columns to make them adopt a particular feature.

Decorations are included within a "Name-Value Pair" storage format, here are some examples:

```
[Properties]
SecurityLevelEdit=50
SecurityLevelInsert=80
ReuseLast=1
Distinct=1
```

In the above example, only users with a "SecurityLevel" above 80 can add a new record to the table, users must have a "SecurityLevel" of 50 to edit a record, when a new record is created the system will automatically give this column the same value that was used in the prior data entry and when the user-entry form is created it will have a "distinct values list" added, this is a list which shows all the other values which have been added to this column, and which allows users to pick entries from this list.

If a field only uses a single decoration there is no need to store it as a name value pair. For example the following Definition is acceptable:

```
ADD COLUMN Comment VARCHAR(100) DEFAULT 'No Comment' DESCRIPTION 'ReuseLast'
```

Framework standards: Data-table relationships

Orixa uses the norms of Relational Databases to structure an App. Hopefullly Developers using Orixa will have experience with these, but if not they should review available tutorials on the internet.