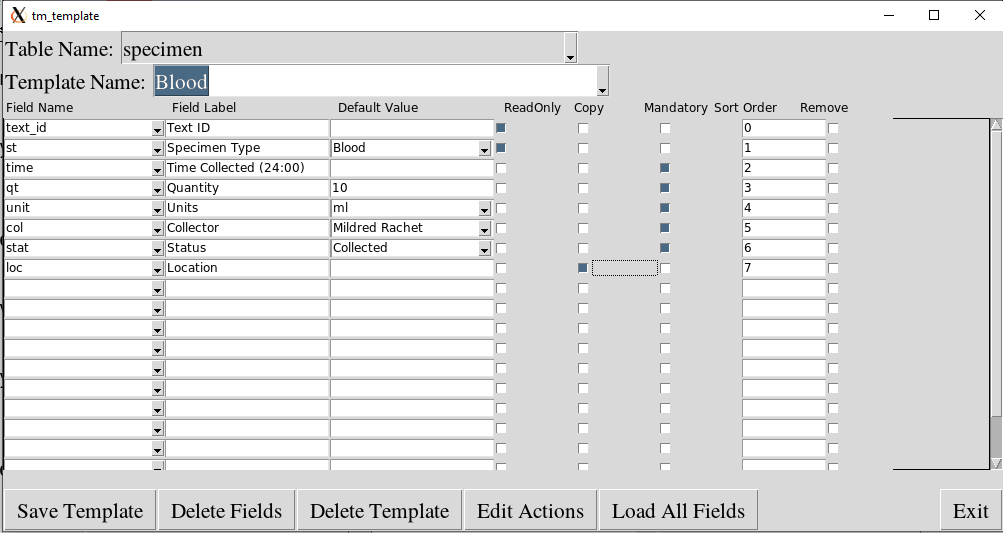
**Database Table Maker v4.0 upgrade notes**

Kenneth Watanabe

**Templates**

Templates make data entry easier by allowing default values to be applied when a record is created. You can also specify which fields are mandatory so that records cannot be created without the specified mandatory fields filled.

To create a template, click on the templates tile. A screen similar to the following will appear. Enter a table name and the fields you wish to prompt for. Clicking on the Load All Fields button will load all the fields of the specified table into the template.



If the ReadOnly checkbox is checked, the field cannot be edited. If the Mandatory box is checked, the user has to enter a value into the field in order to create a new record. If the Copy checkbox is checked, if multiple records are created, then the value of the previously created record will be copied to the next record to be created. The remove checkbox means the field will not be prompted for during data entry.

The Edit Actions button allows access to routines to calculate values for the fields. The following is an example of template actions.

from tm\_increment import \*

import time

class template\_action:

def pre\_entry(self,values):

print("Pre Entry")

values["time"]=time.strftime("%H:%M")

def post\_entry(self,values):

print("Post Entry")

values["text\_id"]="BL-"+time.strftime("%Y")+"-"+values["visit"]+ "."+ increment("visit.identifier",values["visit"])

def post\_create(self,values):

print("Post Create")

The pre\_entry function allows to calculate values prior to entering data. In the above example, the pre\_entry function sets the “time” field to the current time.

The post\_entry function allows to calculate values after the user has entered values and clicked on the Create Record button, but before the record is actually created. In the above example, the “text\_id” field gets set to “BL-“ plus the year, plus the visit number and an incrementing number. For example “BL-2020-5.3” where 5 is the visit identifier and 3 is the third specimen of the visit.

Another function that can be used is the get\_record function. It takes a table\_name, a values dictionary, a record number and a parent record id. The record number can be “LAST, “FIRST” or an integer. The parent record id is the id of the parent record of the table entry. This value should be 0 if the table does not have a parent table. If there is no record, the return value will be false.

For example:

patient\_values = {}

status = get\_record(“patient”,patient\_values,23,0)

This will set the patient\_values to contain all the values of the 23rd patient entered into the database.

visit\_values = {}

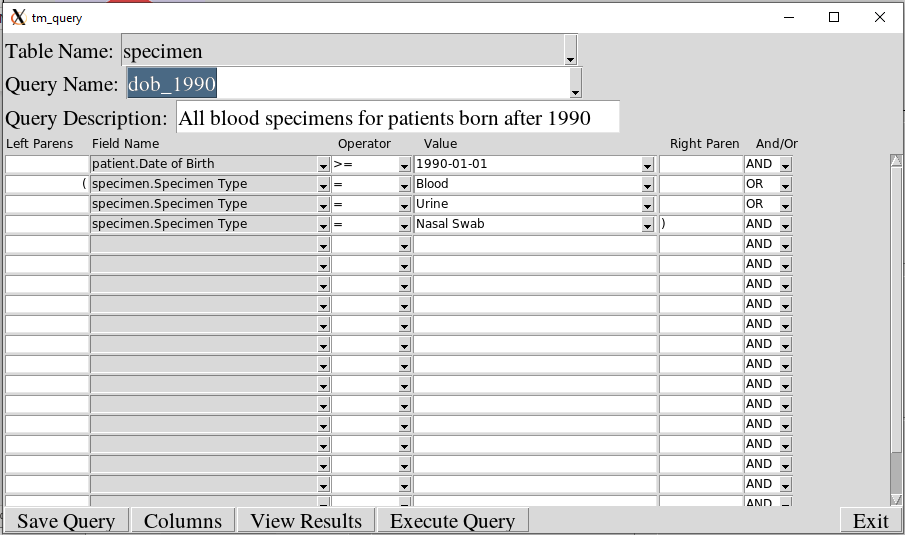
status = get\_record(“vist”,visit\_values,”LAST”,23)

This will set the visit\_values to the last visit for patient 23. If there was no visit for patient 23, then a value of False will be returned and the visit\_values dictionary will be empty.

**Queries**

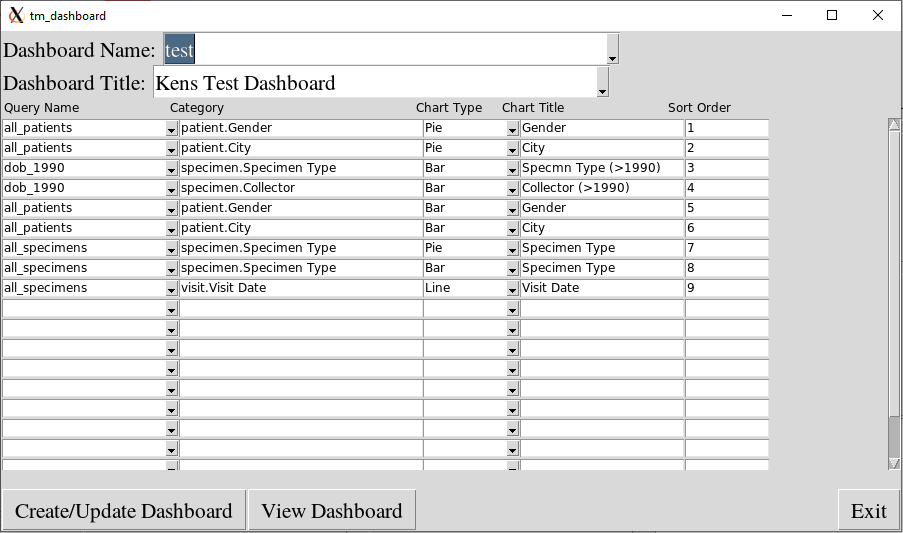
To create a query, click on the Query Data tile from the main menu. A screen similar to the one below will appear. Enter a table name, query name and description. Then enter the criteria for the query. If a child table is entered as the table name, you can query on any field in the child table and any field in the parent table of the child.

In the example below, all specimens of type Blood, Urine or Nasal Swab that belong to patients born after 1990-01-01 will be queried.



**Dashboards**

To create a dashboard, click on the Dashboard tile from the main menu. A screen similar to the one below will appear. Enter a dashboard name and title. Then enter the query names, the category (a column of the query) and the chart type (pie, bar or line).



Save the dashboard and click on the View Dashboard button to view the charts. Below is an example of what a dashboard may look like.

