

**Implementation Activity: A01**

Spring 2013

Due: 05/01 @ 12 noon

## Activity 1 – Workflow Model and API

**Please note:** You may do this activity alone or in groups of two.

**Objective**

The goal of this activity is to develop your knowledge for the relational database model, the PostgreSQL DBMS, and your skills for back-end SQL development.

**Preliminary steps**

The information you need to complete this individual assignment can be found in the *INFO-445 Studio Workbook* on the course website.

- (1) The first step for completing this activity is to install PostgreSQL on your UW student server account or course server:

|                  |                         |
|------------------|-------------------------|
| <i>Machine:</i>  | dante.u.washington.edu  |
| <i>Net ID:</i>   | your UW Net ID          |
| <i>Password:</i> | your UW Net ID password |

If for some reason you are unable to install PostgreSQL on your UW student server account or other server you can use this database which has been created for you:

|                     |                        |
|---------------------|------------------------|
| <i>Machine:</i>     | homer.u.washington.edu |
| <i>Port:</i>        | 45678                  |
| <i>DB User ID:</i>  | your UW Net ID         |
| <i>DB password:</i> | pass                   |

- (2) The second step is to review the topics in the workbook. Once you've done these two things you are ready to begin.

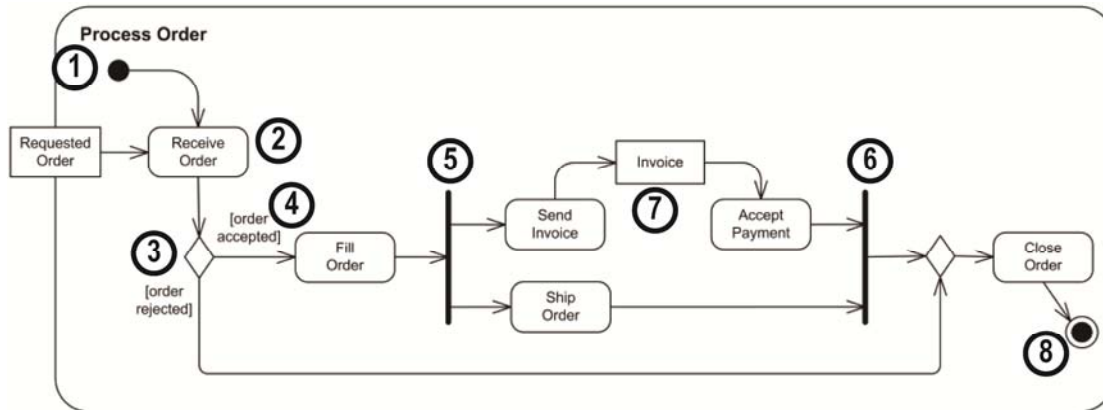
**Implementation Activity: A01**

Spring 2013

Due: 05/01 @ 12 noon

**Conceptual model**

In many situations it is useful to be able to computationally model a workflow. The following is an example of a workflow, modeled as an activity diagram in UML.



**Figure 1.** An example workflow model (taken from [www.uml-diagrams.org](http://www.uml-diagrams.org)).

This notation for representing activity diagrams consists of the following symbols:

|   | Symbol           | Meaning                                                 |
|---|------------------|---------------------------------------------------------|
| 1 | Start            | The starting point for the workflow                     |
| 2 | Activity         | An activity to be completed                             |
| 3 | Decision diamond | A choice must be made or flow of control comes together |
| 4 | Guard label      | A rule about which path to follow                       |
| 5 | Fork             | The workflow is to follow two paths simultaneously      |
| 6 | Join             | Two or more concurrent paths back into one path         |
| 7 | Document         | A document is passed along in a workflow                |
| 8 | Finish           | The finishing point for the workflow                    |

**Implementation aim**

You are to implement a front-end and back-end API for creating and working with workflows.

*Note:* You will likely reuse this API in your class project, so good work on A01 will pay off on your project.

**Implementation Activity: A01**

Spring 2013

Due: 05/01 @ 12 noon

**Front-end scripting API**

The front-end API will implement the scripting commands shown in table 1. You will implement these functions by extending a PHP framework, available on the course website. Conceptually, these commands will provide an API for implementing many different kinds of workflow applications.

**Table 1.** Front-end workflow scripting commands. These commands use a number of standard parameters and data types (see Table 2).

| Command                                                                                          | Meaning                                                                                                                                                                                                                                                                     |
|--------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| workflow create -n <workflow name><br>-i <info>                                                  | Creates a workflow named <workflow name>, including a short informational message <info>. When a workflow is created a start and a finish node are automatically added to the workflow.                                                                                     |
| workflow delete -n <workflow name>                                                               | Deletes the workflow named <workflow name>. When a workflow is deleted all nodes and links are also deleted.                                                                                                                                                                |
| workflow list                                                                                    | Lists all workflows that have been created.                                                                                                                                                                                                                                 |
| node add -wf <workflow name><br>-sn <node short name><br>-t <node type><br>-n <node name>        | Add a node named <node short name> to the workflow <workflow name>. The node type can be either "A" for activity node; "F" for fork node; or "J" for joiner node; "S" for starting node; or "E" for finishing node. The <node name> is a human-readable label for the node. |
| node list -wf <name>                                                                             | List all nodes in a given workflow.                                                                                                                                                                                                                                         |
| link start -wf <workflow name><br>-to <node short name><br>-g <guard label>                      | Link the start node to a given node named <node short name> with the guard <guard label>.                                                                                                                                                                                   |
| link finish -wf <workflow name><br>-from <node short name><br>-g <guard label>                   | Link a given node named <node short name> to the finish node with the guard <guard label>.                                                                                                                                                                                  |
| link -wf <workflow name><br>-from <node short name><br>-to <node short name><br>-g <guard label> | Link two nodes with the guard <guard label>.                                                                                                                                                                                                                                |
| link children -wf <workflow name><br>-sn <node short name>                                       | List all the children nodes and link information for a node, identified by <node short name>.                                                                                                                                                                               |

**Table 2.** Parameters and data types for the workflow scripting commands

| Parameter         | Data type                               |
|-------------------|-----------------------------------------|
| <workflow name>   | A short string without spaces           |
| <info>            | A text field with spaces                |
| <node short name> | A three character string without spaces |
| <node type>       | A domain comprising {A,F,J,S,E}         |
| <node name>       | A string with spaces                    |
| <guard label>     | A string with spaces                    |

**Implementation Activity: A01**

Spring 2013

Due: 05/01 @ 12 noon

**Back-end API**

To implement the above scripting commands you will need to implement a back-end API. The back-end API will implement the functions shown in Table 3. To design and implement this back-end API you will need to do three things:

- (1) Design a database model for modeling the workflow data (workflows, nodes, and links).
- (2) Write **create table** statements to implement the database model.
- (3) Design, implement, and install a set of PL/pgSQL functions into your database.

Once these functions are implemented you will be able to create a manage workflows as demonstrated by this very simple SQL script:

```
// Create a workflow
select dtw.Create_workflow('work', 'Simple test of the workflow module');

// Add three nodes to the workflow
select dtw.Add_node('work', 'A', 'Document submitted', 'A');
select dtw.Add_node('work', 'B', 'Document being reviewed', 'A');
select dtw.Add_node('work', 'C', 'Document approved', 'A');

// Link up the nodes
select dtw.Link_from_start('work', 'A', '');
select dtw.Link_between('work', 'A', 'B', 'Need to assign document');
select dtw.Link_between('work', 'B', 'C', 'Need to approve document');
select dtw.Link_to_finish('work', 'C', '');

// Get the children of the workflow
select dtw.Get_children('work', 'A');

// Access information about the workflow nodes
select dtw.Get_node_by_id(233);
select dtw.Get_node('A');

// Drop the workflow
select dtw.Drop_workflow('work');
```

**Implementation Activity: A01**

Spring 2013

Due: 05/01 @ 12 noon

**Table 3.** Back-end PL/pgSQL functions.

| <b>PL / pgSQL Function</b> | <b>Purpose</b>                                                                              |
|----------------------------|---------------------------------------------------------------------------------------------|
| Create_workflow ()         | Used to create an empty workflow.                                                           |
| Drop_workflow ()           | Used to delete a workflow.                                                                  |
| Get_workflows ()           | Used to get a list of all workflows that have been created.                                 |
| Add_node ()                | Used to add a new node to a workflow.                                                       |
| Get_nodes ()               | Used to get a list of all nodes in a workflow.                                              |
| Link_between ()            | Used to add an link between two nodes                                                       |
| Link_from_start ()         | Used to link from the special start node to the first node in the workflow                  |
| Link_to_finish ()          | Used to link from the last node in the workflow to the special finish node in the workflow. |
| Get_children ()            | Used to get information on the children of a node.                                          |

Notes: (1) The parameters for each of these functions are not shown; (2) You may decide to implement additional functions.

**Hooking up the front-end to the back-end**

Once you've implemented the back-end (or at least some of the back-end functions) you will need to hook up the front-end PHP scripting framework to the back-end API. Once you've done this you will be able to test your backend code and implement a range of front-end applications.

**Implementation Activity: A01**

Spring 2013

Due: 05/01 @ 12 noon

**The deliverables**

You should submit a PDF report and a website. Aim for conciseness, precision, and clarity.

**The Report Structure: What to Include?**

**0.0 Title page** (page 1). Include a title, your name(s), link to your website, etc.

**1.0 Introduction** (page 2). You briefly introduce the goals of this project in your own words. Target this writing to an employer who knows about data systems but not this particular assignment.

**2.0 Architecture** (page 3). Present a drawing of the three-tier architecture for your system. Include as much detail as possible, including:

- a. All the technologies that you are using at the front-end, the back-end, and in the middle tiers;
- b. How the software you have developed is divided into layers at the front-end and the back-end;
- c. How the front-end and the back-end communicate;
- d. How the architecture could be extended by other developers.

Include a caption that concisely describes the architecture.

**3.0 Logical model** (page 4). Present the logical model for your system. Include a caption that concisely describes the model.

**4.0 Discussion** (page 5-6). (a) Summarize the progress that you've made on this project; (b) Briefly discuss any limitations; and (c) Provide some brief reflections on what you have learned.

**The Website: What to Include?**

1. A simple title page to the assignment
2. A link to the report
3. A link to the command shell that can be used to test your application
4. A clear demonstration of your implementation
5. Links to all code used in your application.

**Grading rubric**

1. Overall neatness and attention to detail; Absence of spelling and grammatical errors; Clear organization and concise writing and visual expression.
2. All diagrams are clear and include a full caption.
3. The PHP and SQL code is clearly written and included on the website.
4. The ER model shows the entities, attributes, attribute types, relationships, and cardinality and participation constraints. The model uses a standard ER modeling notation *rigorously* and correctly. The model does *not* contain M-M relationships, generalization/specialization relationships, or ternary relationships.
5. The three-tier system architecture shows all technologies used and the structure of the system.
6. The front-end and back-end APIs are implemented as specified and you provide a simple, efficient way to check the implementation. The PHP and SQL code is easily accessed.