## Problem Description

A user needs to be able to drill down into a tree of hierarchical data and when selecting the terminating node in this tree will see some detail of the children of this collection in a table to its right and additional detail of the selected row in a form below.

## Technical Pattern Description

The Master-Detail-MoreDetail pattern provides an ADF Tree to display the Master information, an ADF Table showing some Detail of the children of the terminating node selected in the tree and an ADF Form showing more detail of the item selected in the Table. See the User Experience section below for a screenshot of the user interface from the pattern prototype.
The User Experience

Figure 1: Master-Detail-MoreDetail UI

Step 1: Drill down into the Tree and select the desired terminating node (Departments, in this case)

When the user selects the desired Department, the Table to its right will refresh to display the Employees for this Department, showing some detail of each Employee.

Step 2: View more detail of target Employee

Now the user can select the desired Employee from the Table and see additional detail information for this employee in the Form below.
The Artifacts

Following is a list of the artifacts used in this pattern.

ADF BC Model Objects

- **RegionsView1**: The source VO for the root node in the Tree. Via hierarchical view accessors (created per FK links), the tree also access data from the CountriesView, LocationsView, and DepartmentsView via the RegionsView1Iterator.

- **DepartmentsView1 & EmployeeView1**: The Table and Form access Employee data using the EmployeesView1 view instance created under DepartmentView1 via the EmpDeptFKLink.

User Interface Elements

- **ADF Tree**: This UI component is based on the RegionsView1 Collection.
- **ADF Read Only Table**: This UI component is based on the EmployeesView1 collection located under the DepartmentsView1 Collection.
- **ADF Form**: This UI component is based on the EmployeesView1 collection located under the DepartmentsView1 Collection.

Implementing the Pattern

Note: This sample implementation of this pattern uses the HR schema in the oracle demo database.

The implementation of the Master-Detail-MoreDetail pattern can be broken down into 3 fundamental elements:

- Create the Application, Business Components and Application Module
- Create the UI
- Hook up the Tree to the Table and enable Partial Refresh

Note: Unless otherwise instructed, accept the ADF Wizard defaults options.

**Create the Application, Business Components and Application Module**

1. Make sure that your Roles Preference is set to the ‘Default Role: This will enable all technologies. Go to Tools > Preferences and select the Roles item.
2. Create a new **Fusion Web Application (ADF)**.

3. Select the Model project and create **ADF Business Components** using the ‘Business Components from Tables’ wizard.
   a. Create a new connection to your HR demo database
   b. If Auto-Query is not enabled, press [Query]
   c. Step 1 (EO): Move the following tables from the Available list into the Selected Lists: COUNTRIES, DEPARTMENTS, EMPLOYEES, LOCATIONS, and REGIONS.
   d. Step 2 (Updatable VO): Move all the Available VO into the Selected List.
   e. Step 3 (RO VO): Skip
   g. Step 5 (Diagram): Skip
   h. Step 6 (Summary): Click [Finish].

4. Create an **Application Module** for your Model project and put the following View Objects into its Data Model: DepartmentsView & RegionsView. Now, select the DepartmentsView1 item in your Data Model list then move over the "EmployeesView via EmpDeptFkLink" from the list of Available Data Objects.

5. In this example, we want our Tree (created below) to only display nodes that include rows in the leaf node (i.e. Departments). In order to do this, we will add where clauses to the RegionsView, CountriesView and LocationsView by doing the following:

   Master-Detail-MoreDetail
Functional

a. Expand the Model Project, then expand the Application Sources and model nodes. Now select the desired view (RegionsView, CountriesView and LocationsView).

b. Select the Query page, then click the Query section’s [edit] icon.

c. Now enter the where clause (see data below) and click [OK].

Use the following where causes…

RegionsView where clause:
region_id in (select region_id
from countries C, locations L, departments D
where C.country_id = L.country_id and
L.location_id = D.location_id)

CountriesView where clause:
country_id in (select country_id
from locations L, departments D
where L.location_id = D.location_id)

LocationsView where clause:
location_id in (select location_id from departments)
Functional

Create the UI

1. Prepare your **Page** for your Tree-Table-Form as follows:
   
d. Collapse your Model Project, expand the View Controller Project and Open adfc-config.xml.

e. Drag-and-Drop a View component from the Component Palette onto your adfc-config.xml window. Give this new view a name and create the page by double-clicking the view that you just created.

f. Drag-and-Drop a Panel Splitter component from the Component Palette onto your Page.

![Figure 3: Component Palette –Panel Splitter](image)

   g. Select the ‘second’ panel on your page and then drag-and-drop another Panel Splitter component onto the ‘second’ panel. In the Property Inspector, change the Orientation Property to ‘vertical’.

![Figure 4: Property Inspector –Panel Splitter Orientation Property](image)
2. Add your master Tree to the Page as follows:
   a. From the Data Controls Panel, drag-and-drop the RegionsView1 Collection over to the left-most ‘first’ panel and choose to create an ADF Tree (Tree>ADF Tree).
   b. On the Edit Tree Binding dialog, notice that RegionsView1 has been automatically selected as the Root data source of the Tree and the first node (aka. Tree Level Rule) of the Tree.
   c. Change the Display Attributes to show RegionName instead of the defaulted RegionId.
   d. Select [+] to add the next tree level rule (CountriesView). (Note: CountriesView shows up here, because there is an accessor, created automatically, on the RegionsView1 for the CountriesView.) Change the Display Attributes to show CountryName instead of the defaulted CountryId.
Functional

e. Select the [+] again, this time to add the LocationsView node. Change the Display Attributes to show City instead of the defaulted LocationId.

f. Select the [+] again, to add the DepartmentsView node. Change the Display Attributes to show DepartmentName instead of the defaulted DepartmentId. (Note: A Target Data Source needs to be set for this node, but first we have to create it.)

Figure 6: Edit Tree Binding
g. After clicking [OK] on the Edit Tree Binding dialog, set the Tree’s id to ‘tree1’ using the Property Inspector. (This id will be used later when we setup Partial Refresh.)

![Property Inspector - Tree Id Property](image1)

*Figure 7: Property Inspector – Tree Id Property*

3. Add your detail **Table** to the Page as follows:
   a. Expand DepartmentsView1 Collection and drag-and-drop the EmployeesView1 Collection over to the right most ‘first’ panel and choose to create an ADF Read-only Table (Tables>ADF Read-only Table). Enable ‘Row Selection’.
   b. Remove the following columns from the Table: HireDate, JobId, Salary, CommissionPct, ManagerId.

![Edit Table Columns](image2)

*Figure 8: Edit Table Columns*

4. Add your more-detail **Form** to the Page as follows:
   a. Expand DepartmentsView1 Collection and drag-and-drop the EmployeesView1 Collection over to the right most ‘second’ panel and choose to create an ADF Form (Forms>ADF Form). Enable ‘Include Navigation Controls’ & ‘Include Submit Button’.
Hook up the Tree to the Table and enable Partial Refresh

Note: If you run your page at this point, the Tree will expand properly, but the Employee data in the Tree (and Form) will not change when you select a Department node. For example, select the Finance Department under Americas > United States of America > Seattle. This is because we have not setup a Target Data Source for the Tree or setup Partial Refresh using Table’s PartialTriggers property.

1. Set the Target Data Source on the Tree’s Departments node by doing the following:
   a. On your page, select the [Bindings] tab.

   ![Figure 9: Bindings Tab (jspx page)]

   b. Create the DepartmentsView1Iterator by doing the following:
      i. Click the [+ Add icon on the list of Executables.

      ![Figure 10: Executables –Add icon]

      ii. Select ‘iterator’ and click [OK].

      ![Figure 11: Insert Item dialog (iterator selected)]

      iii. Expand AppModuleDataControl, select DepartmentsView1 and click [OK]. This creates the DepartmentsView1Iterator that we will use with the Tree’s Department Node.

      ![Figure 12: Executables (with newly created DepartmentView1Iterator)]
c. Under the list of Bindings for your page, select RegionsView1 and press the Edit icon. (The Edit Tree Bindings dialog will appear.)

![Model]

Figure 13: Bindings – Edit Icon

d. Select the DepartmentsView node and expand the Target Data Source section of the dialog.

e. Press [El Picker].

f. Now expand ADF Bindings > bindings and select the DepartmentsView1Iterator and press [OK].
2. Setup Partial refresh by doing the following:
   a. On your page, select the [Design] view and select the Table component.
      (You can also select the Table using the Structure panel.)
   b. Expand the Behavior section of the Property Inspector and locate the
      Partial Triggers property. Expand the drop down for this field and select
      ‘Edit’.
Functional

Setting the Table’s PartialTriggers Property to the Tree’s Id will create a dependency between the Table and the Tree. If the Tree receives an event that will cause it to update, the Table will request to be updated too.

c. On the Edit Property dialog, expand the first ‘facet (first)’ item and select ‘tree – tree1’, then press [>] to shuttle this component to the Selected list.

![Edit Property Dialog](Image)

* indicates a naming container

Figure 15: Edit Property Dialog
Functional

d. Now press [OK] on the Edit Property dialog and notice that ‘::tree1’ was inserted into the PartialTriggers Property.

![Property Inspector – Table’s PartialTriggers Property](image)

**Figure 13: Property Inspector – Table’s PartialTriggers Property**
3. Related Documentation

a) Fusion Developer's Guide for Oracle ADF
b) Web UI Developer's Guide for Oracle ADF

<table>
<thead>
<tr>
<th>Documentation Related to the Pattern Artifacts</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>ADF Business Components</td>
<td>Creating Entity Objects and Associations in the Fusion Developer’s Guide for Oracle ADF</td>
</tr>
<tr>
<td>Application Module</td>
<td>Creating and Modifying an Application Module in the Fusion Developer’s Guide for Oracle ADF</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Documentation Related to Pattern Implementation Steps</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Controls Panel</td>
<td>Using the Data Controls Panel in the Fusion Developer’s Guide for Oracle ADF</td>
</tr>
<tr>
<td>ADF Tree</td>
<td>Displaying Data in Trees in the Web UI Developer’s Guide for Oracle ADF</td>
</tr>
<tr>
<td>ADF Table</td>
<td>Displaying Data in Tables in the Web UI Developer’s Guide for Oracle ADF</td>
</tr>
<tr>
<td>Partial Page Refresh</td>
<td>Enabling Partial Page Rendering Declaratively in the Web UI Developer’s Guide for Oracle ADF</td>
</tr>
</tbody>
</table>

4. Appendix – Object Definitions Used by the Prototype

In order to test this pattern out, included is a JDeveloper 11g Application called master-detail-moredetail that can be unzipped into your designated work area and explored / executed from within JDeveloper. Simply download master-detail-moredetail.zip, unzip the archive, open the mdmd.jws with JDeveloper 11g and explore.