Penny's Portal Techniques – Accessing the Repository

Introduction

There are a number of reasons why it may be necessary to access the underlying Oracle Portal repository tables. While updating the repository directly is not supported, queries on the repository can be used for creating dynamic portlets. This article provides information on the structure of some of the most useful Oracle Portal tables, and their relationships. It contains some simple examples of using the portal tables to create dynamic portlets.

Core Tables

Figure 1 displays a diagram of some useful repository tables as follows:

wwv_things	This table contains a row for each content item in the portal.
wwsbr_sites$	This table contains a row for each page group.
wwv_topics	This table contains a row for each category.
wwv_perspectives	This table contains a row for each perspective.
wwv_thingsperspectives	This table contains the link between items and perspectives.
wwpob_page$	This table contains a row for each page in the portal.

The columns that are required to join the tables are displayed on the relationship lines. Note that these are not necessarily implemented as foreign key constraints in the database, rather this is the information that is required to join the tables in most practical situations. I have ignored language information, and assumed that you are working in a single language portal. To extend the model to cater for multiple languages you will need to consider the WWSBR_SITE_LANGUAGES$ table.

The two recursive relationships on the WWPOB_PAGE$ table are as follows:

WWV_CORNERS_PARENT_FK

SUBSCRIBER_ID	references	WWPOB_PAGE$.SUBSCRIBER_ID
PARENTID	references	WWPOB_PAGE$.ID
SITEID	references	WWPOB_PAGE$.SITEID
LANGUAGE	references	WWPOB_PAGE$.LANGUAGE

This represents the hierarchical structure of the pages. The foreign key allows you to perform a tree walk to obtain a page's friendly url.

WWV_CORNERS_TEMPLATE_FK

SUBSCRIBER_ID	references	WWPOB_PAGE$.SUBSCRIBER_ID
TEMPLATE_ID	references	WWPOB_PAGE$.ID
TEMPLATE_SITEID	references	WWPOB_PAGE$.SITEID
LANGUAGE	references	WWPOB_PAGE$.LANGUAGE

This records the template that is used for a page. It can be used to identify pages which have been based on a template as follows.

Example 1: List pages with no template

```
SELECT id, name
FROM portal.wwpob_page$
WHERE template_id IS NULL
```

Example 2: List pages using a particular template

```
SELECT    s.name, p.id,p.name
FROM      portal.wwpob_page$ p, portal.wwsbr_sites$ s,
          portal.wwsbr_sites$ ts, portal.wwpob_page$ t
WHERE     t.id = p.template_id
AND       p.siteid = s.id
AND       t.subscriber_id = p.subscriber_id
AND       s.subscriber_id = p.subscriber_id
AND       s.subscriber_id = ts.subscriber_id
AND       t.siteid    = ts.id
AND       ts.name = '&TEMPLATE_PAGE_GROUP'
AND       t.name   = '&TEMPLATE_NAME'
```

Security

Issues

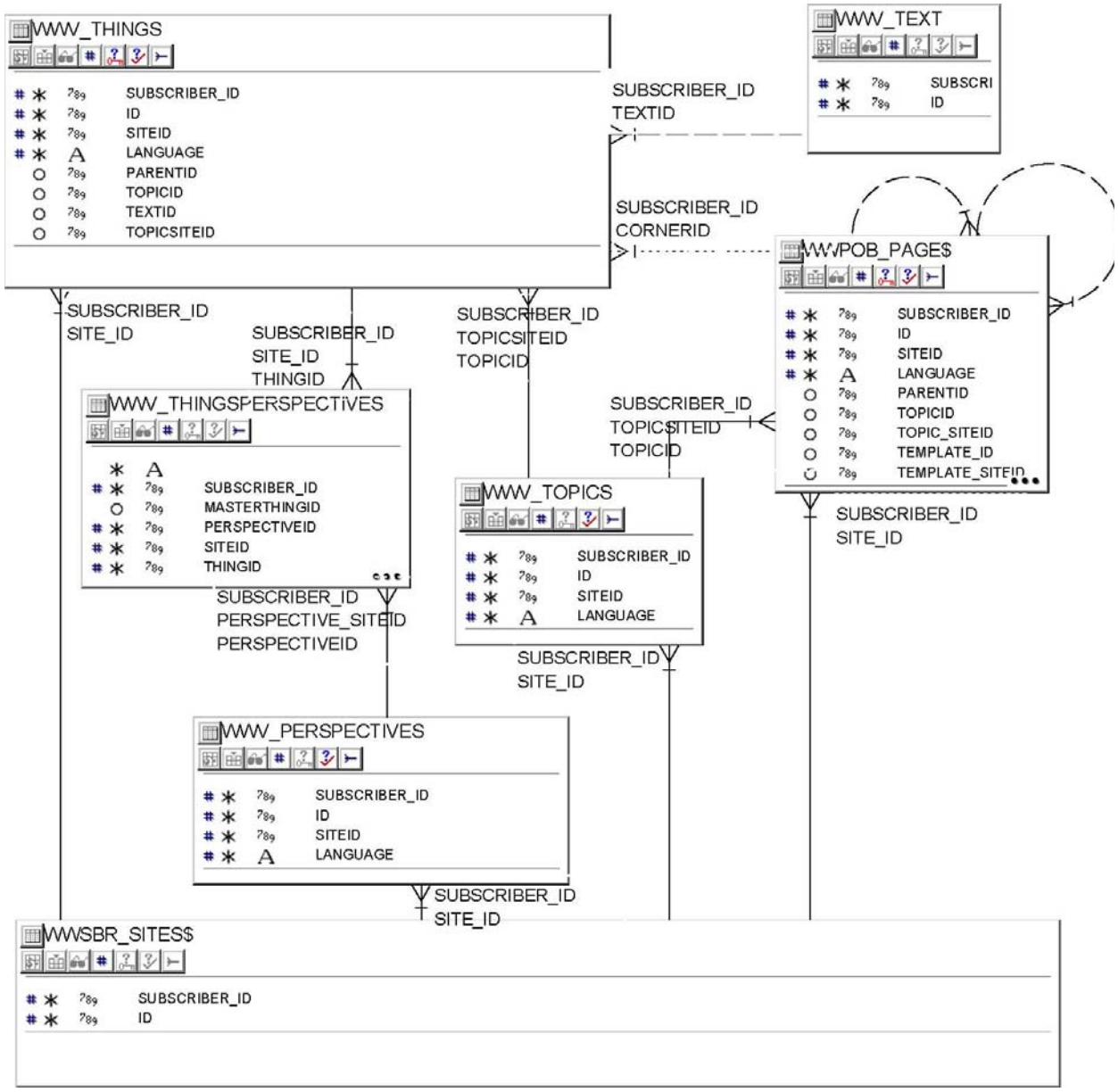
It is important to note that when you access the portal tables directly you are bypassing any security stored in Oracle Portal. As an example, if Page1 and Page2 have public access, while Page3 is only accessible to users in the ADMIN group, if you select from WWPOB_PAGE$ then all pages are available.

Views v Tables

Oracle provides a series of views that implement the Portal security. If the information in your repository is not public, then you should use these views rather than accessing the tables directly. You may need to consider creating your own views to implement security in some cases.

Example:
WWPOB_PAGE$ accesses all pages
WWV_CORNERS accesses pages to which the portal user has at least View privileges.

Figure 1 – Core Tables



Storing Images

It is recommended that you create a page within each page group specifically for images. Store all images for the page group in this page.

This technique also allows you to dynamically display images based on a set of parameters. You can assign a standard category to an image, for example BANNER_LOGO. This category can then be used to select the image url from the underlying portal tables.

The following code accepts parameters that provide a Category and a Page Group. It will then return the url to display the image having that category in the given page group.

Example 3: Get an image url given the category

```
-- used to display standard images in the banner
FUNCTION image_url (p_topic IN wwv_topics.name%TYPE,
                    p_page_group IN wwpob_page$.name%TYPE)
RETURN VARCHAR2
IS
  v_url    VARCHAR2(1000);
  v_image  wwv_things.image%TYPE;

  CURSOR c_image (p_page_group IN wwpob_page$.name%TYPE) IS
  SELECT t.image
  FROM   wwv_things t, wwsbr_sites$ s, wwv_topics c
  WHERE  t.siteid   = s.id
  AND    s.name     = p_page_group
  AND    t.topicid  = c.id
  AND    c.name     = p_topic;

BEGIN

  OPEN  c_image (p_page_group);
  FETCH c_image INTO v_image;
  CLOSE c_image;

  IF v_image IS NULL THEN
    OPEN  c_image ('DEFAULTPAGEGROUP');
    FETCH c_image INTO v_image;
    CLOSE c_image;
  END IF;

    v_url := wwctx_api.get_direct_doc_path(p_name=> v_image);

  RETURN v_url;

END image_url;
```

Current News Portlet

The following code can be use to create a simple “Current News” portlet. Items which are considered to be current are allocated a CURRENT_NEWS perspective. When items are no longer current they do not need to be moved or deleted, they simply have the perspective deallocated.

N.B. To simplify this example, this code assumes that there is only a single level of pages in the page group. The example could easily be expanded to adapt to a more complex page structure.

The example accepts two parameters, perspective and page group and displays items that are in a particular page group, and which have been assigned a given perspective.

Note that I have left the subscriber_id out of the code to simplify it. (It will almost always be 1)

This procedure can be called from a dynamic page as follows:

```
<ORACLE>
BEGIN
train1.current_news(:p_page_group,:p_perspective);
END;
</ORACLE>
```

The perspective and page group can be passed in as a parameter.

The portlet uses the api packaged procedure portal.wwui_api_portlet.portlet_text to render the item. This ensures that the style of the item matches the style of the page on which the portlet is displayed.

The link provided will display the page that the item is on, rather than the item itself without the page context.

Example 4: Use the portal tables to display items with the current news perspective

```
CREATE OR REPLACE PROCEDURE current_news
 (p_page_group IN portal.wwv_user_corners.name%TYPE,
  p_perspective IN portal.wwv_perspectives.name%TYPE := 'CURRENT_NEWS')  IS

CURSOR c_news IS
SELECT t.title, t.guid,  t.description, pg.name page_name, s.name page_group_name
FROM   portal.wwv_things t, portal.wwv_thingsperspectives tp,
          portal.wwv_perspectives p, portal.wwsbr_sites$ s, portal.wwpob_page$ pg
WHERE  t.siteid = s.id
AND    s.name = p_page_group
AND    t.id = tp.thingid
AND    t.siteid = tp.siteid
AND    tp.perspectiveid = p.id
AND    upper(p.name) = UPPER(p_perspective)
AND    pg.id = t.cornerid
AND    pg.siteid = s.id
ORDER BY t.createdate desc;

  v_page_url  VARCHAR2(4000);

BEGIN

  htp.p('<table width="95% border="0" cellspacing="2" cellpadding="0">');
 -- Print the news items
  FOR r_news IN c_news LOOP

    -- get the item pages
    v_page_url := 'URL/PAGE/'||
                        r_news.page_group_name||'/'||r_news.page_name;

    htp.p('<tr><td><a title='||r_news.title||' href="'||v_page_url||'" >'||
           portal.wwui_api_portlet.portlet_text(nvl(r_news.title,'News'),3)||
           '</a></td></tr>' );

    IF r_news.description IS NOT NULL THEN
      htp.p('<tr><td><p align="Justify">'||
      portal.wwui_api_portlet.portlet_text(r_news.description,4)||
       '</p></td></tr>');
    ELSE
      htp.p('<tr><td></td></tr>');
    END IF;

    htp.p('<tr><td></td></tr>');
  END LOOP;
  htp.p('</table>');

END current_news;
```