

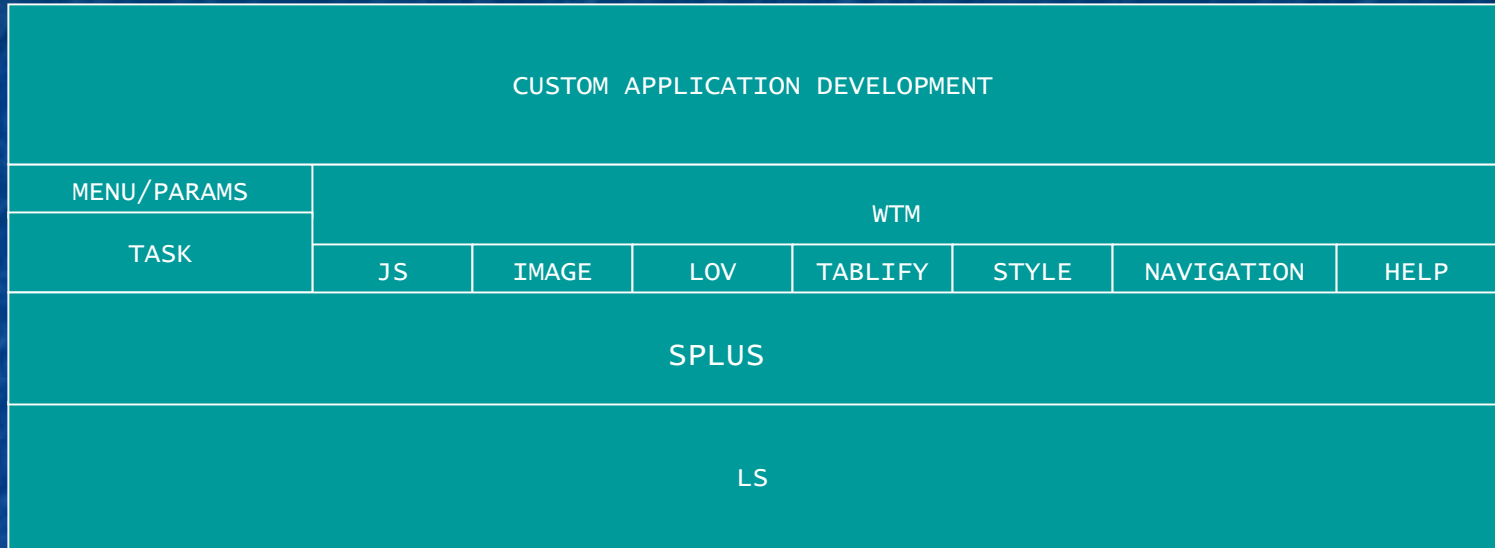
A Hierarchical File System for PL/SQL

Presented by: David Bibeau
Project Manager
Dartmouth College

What is it?

- A way to create/store and maintain documents (content) within Oracle
- API Accessible from PL/SQL
- Web based user interface
- Bulk Load/Unload from command line Java client
- Part of a broader framework developed at Dartmouth College for HTTP based apps

Admin Computing In-House Development (ACID) Framework



- WTM - Web Table Maintenance, builds default forms for Oracle tables
- Menu/Params - Standard Menu/Param prompting facility
- TASK - Task subsystem for executing SQL in an envelope that allows scheduling, cancelling, auditing
- JS - Javascript Manager
- IMAGE - Image Manager
- LOV - List Of Values Manager
- TABLIFY - Text Loader allows client files to be uploaded to the database
- STYLE - CSS Style Sheet Manager
- NAVIGATION - Navigation provides scrolling through SQL resultset
- HELP - Help System Manager
- SPLUS - SQLPlus Interpreter implemented in PL/SQL, basic report generator
- LS - LOB System, provides a pseudo filesystem for storing data in Oracle

What is it? Cont.

Open Source on sourceforge.net

- <http://sourceforge.net/projects/pldev>
- LS_API and PDF packages originally developed by Gary Menchen
- Additional contributions by David Ricker

Quick Demo of UI

File Explorer - Mozilla Firefox

File Edit View Go Bookmarks Tools Help

https://oracle-dev.dartmouth.edu:8443/dart/aharpo/eic.filesystem.directory?working_path=/users/OPS\$ADP_BIBEAU

ZINC Endowment Fl... IRON Endowment Fl... COPPER Endowmen... Replicon Web TimeS... Project Planning Sub with Bloglines

File Explorer SourceForge.net CVS Repository - director...

Database: ZINC
User: OPS\$ADP_BIBEAU

Directory of Files Date: 11/22/2005 10:39:11

Owner Filter: % File Filter: % Refresh Purge

Up Cmd

All Folders

- /
- users
- OPS\$ADP_BIBEAU

File Name (asc)	K	Create Date
DBM_LS_USAGE_12816.PDF	222.1	10/18/2005
ED_BUDGET_PROJECTION_13275.PDF	1.0	10/31/2005
EFS_LIT_UPLOAD_CORRECTION.txt	32.7	08/10/2005
EFS_UPLOAD_20050729.txt	26.3	08/02/2005
EFS_UPLOAD_20050802.txt	26.1	08/02/2005
EFS_UPLOAD_20050802_LIABILITY.txt	16.2	08/02/2005
FASB_REVIEW_12402.PDF	332.7	10/05/2005
FASB_REVIEW_12403.PDF	332.6	10/05/2005
FUND_BALANCE_69670_10046.PDF	26.9	08/08/2005
FUND_MOD_DETAIL_12372.PDF	25.1	10/05/2005
Fund_Mult_and_Resid_Value_Sensitivity_10364.xls	1,396.3	08/15/2005
FUND_PRE_CLOSE.PDF	2.4	08/11/2005
FUND_PRE_CLOSE_10257.PDF	11.0	08/11/2005
FUND_PRE_CLOSE_13304.PDF	2.4	11/04/2005
FUND_PRE_CLOSE_13307.PDF	10.9	11/04/2005
FUND_TRANSACTIONS_9807.PDF	3.3	08/04/2005
FUND_TRANSACTIONS_9808.XLS	43.6	08/04/2005
HOLDINGS_BY_CASPIR_ACCOUNT_13315.PDF	523.2	11/07/2005
HOLDINGS_BY_CASPIR_ACCOUNT_13316.PDF	546.1	11/07/2005
HOLDINGS_BY_CASPIR_ACCOUNT_13317.PDF	546.2	11/07/2005

Directory: OPS\$ADP_BIBEAU

- [Privs](#)
- [Rename](#)
- [Delete](#)

My Directory Task Files SQL Blog Menu Home

New Directory New File Upload Command Up Multi-File Explorer

Done oracle-dev.dartmouth.edu:8443

Motivation

- Did not want to re-engineer hundreds of reports in something like Reports 6i
- Needed a place to store execution scripts, logs and other generated output “inside” the database (i.e. no network drives, client desktops etc.)
- Looked into Oracle iFS, but it is not accessible from PL/SQL
- The delivered WebUI uses the LS_API for storing external images and javascript

Installation

- Requires Oracle database version 9i
- Web UI requires iAS with mod_plsql (HTTP toolkit)
- Runs in a minimum priv environment (i.e. does not need DBA privs) but does require RESOURCE (create tables, packages etc) and CREATE VIEW

Installation cont.

- Easy to install on a development database and HTTP Server from Oracle (10g)

```
C:\bibeau\build_filesystem\create_database.sql
1 CREATE TABLESPACE endow_data DATAFILE 'c:\oradb\data\endow_data.dat' size 300M autoextend on next 100M;
2 CREATE TABLESPACE endow_indexes DATAFILE 'c:\oradb\data\endow_indexes.dat' size 150M autoextend on next 50M;
3 CREATE TABLESPACE lob_data DATAFILE 'c:\oradb\data\lob_data.dat' size 300M autoextend on next 200M;
4
5 create user eic identified by temp default tablespace endow_indexes
6 quota unlimited on endow_data quota unlimited on endow_indexes quota unlimited on lob_data;
7
8 grant create session to eic;
9 grant resource to eic;
10 grant create view to eic;
11 grant create public synonym to eic;
12
13 grant execute on dbms_lock to public;
14 grant execute on dbms_pipe to public;
15
16 grant select on v$database to public;
17
18 create view database as
19 select name, database_name from v$database;
20
21 grant select on database to public;
22 grant select on database to eic;
23
```

Installation cont.

■ Run a series of SQL scripts

```
@c:\bibeau\build_filesystem\build_utility.sql
```

```
@c:\bibeau\build_filesystem\build_ls_api.sql
```

```
@c:\bibeau\build_filesystem\build_htp_base.sql
```

```
@c:\bibeau\build_filesystem\build_doc_upload.sql
```

```
@c:\bibeau\build_filesystem\build_filesystem.sql
```

■ Add Ons

```
@c:\bibeau\build_filesystem\build_pdf.sql
```

```
@c:\bibeau\build_filesystem\build_splus.sql
```

Features of Utility Layer

■ CHAR_TO_DATE

- Convert any date string in any format to date

`ex_char_to_date.sql`

■ TRACER

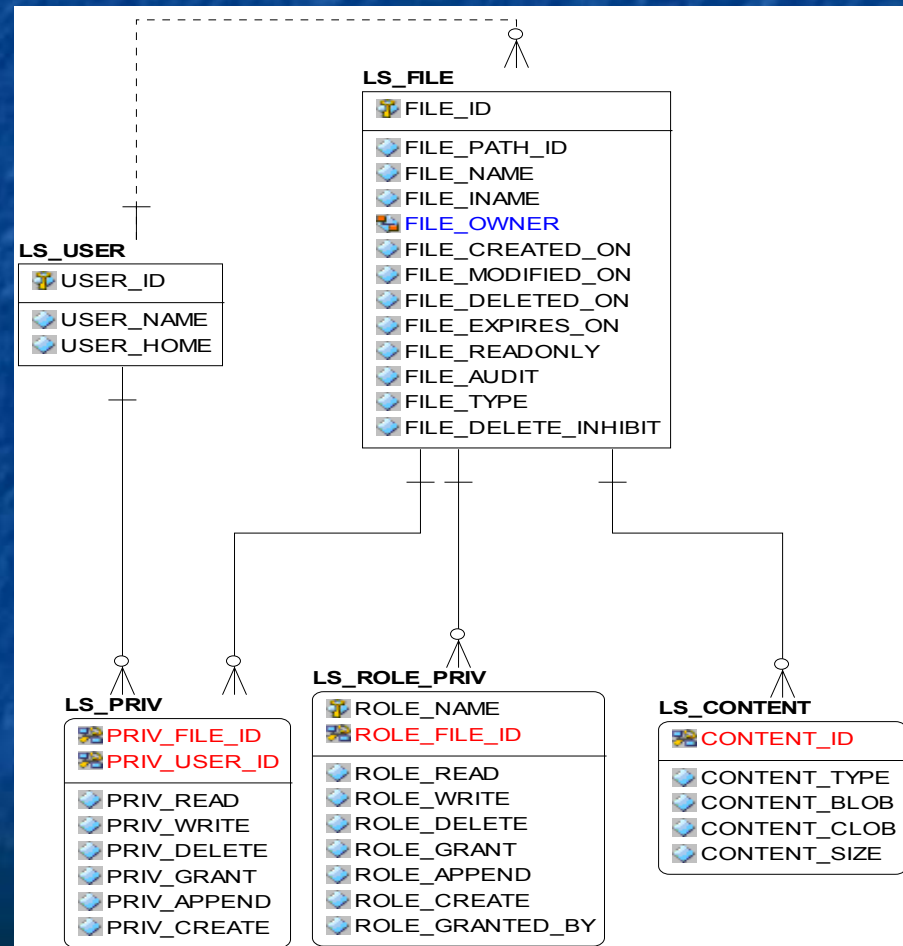
- Debug PL/SQL (anywhere) real time `ex_tracer.sql`

■ QRY Object

- Execute dynamic SQL as an object type (like JDBC rowset processing) `ex_qry.sql`

The LS_API Layer

- ER Diagram
 - 5 basic tables



LS_USER table

The screenshot shows a SQL Editor window titled "SQL Editor - EIC(1) File: LS_USER.sql". The editor contains the following SQL query:

```
SELECT a.user_id, a.user_name, a.user_home  
FROM ls_user a
```

Below the query, a status bar indicates: "[1]: Statement processed in 0.00 sec; see Spool tab".

The results are displayed in a table with the following columns: Row #, USER_ID, USER_NAME, and USER_HOME.

Row #	USER_ID	USER_NAME	USER_HOME
1	46775	EIC	0
2	46778	OPS\$ADP_BIE	46779

At the bottom of the window, the status bar shows "Row 1 of 2" and "Read Only".

LS_FILE table

SQL Editor - EIC(1) File: LS_FILE.sql

SQL Spool

```

SELECT a.file_id, a.file_path_id, a.file_name, a.file_iname,
       a.file_owner, a.file_created_on, a.file_modified_on,
       a.file_deleted_on, a.file_expires_on, a.file_readonly,
       a.file_audit, a.file_type, a.file_delete_inhibit
FROM ls_file a
    
```

Statement processed in 0.02 sec; see Spool tab

Def NRD D N

Row #	FILE_ID	FILE_PATH_ID	FILE_NAME	FILE_INAME	FILE_OWNER	FILE_CREATED_ON	FILE_MODIFIED_ON	FILE_DELETED_ON	FILE_EXPIRES_ON	FILE_READONLY	FILE_AUDIT	FILE_TYPE	FILE_DELETE_INHIBIT
1	0		[root]	[ROOT]	46775	21-Nov-2005 8:41:27			19-Feb-2006 8:41:27	0	0	P	0
2	46776	0	users	USERS	46775	21-Nov-2005 8:41:27	21-Nov-2005 8:41:27		19-Feb-2006 8:41:27	0	0	P	1
3	46777	0	public	PUBLIC	46775	21-Nov-2005 8:41:27			19-Feb-2006 8:41:27	0	0	P	0
4	46779	46776	OPS\$ADP_BIBEAU	OPS\$ADP_BIBEAU	46775	21-Nov-2005 8:42:09	21-Nov-2005 8:42:09		19-Feb-2006 8:42:09	0	0	P	1
5	46780	0	eic	EIC	46775	21-Nov-2005 8:42:10			19-Feb-2006 8:42:10	0	0	P	0

Row 1 of 10 (partial) Read Only

Buffering

I did some quick benchmark experiments writing 10000 80 byte records into a clob using various buffer sizes before a flush (dbms_job.writeappend)

buffer size	time in seconds			
	trial1	trial2	trial3	avg
4K	15.73	16.95	16.98	16.55
7.2K	10.89	9.35	10.35	10.20
8K	10.61	9.91	10.25	10.26
16K	8.95	8.48	7.83	8.42
32K	9.55	9.33	9.69	9.52

the 7.2K size is 10% below 8K, since the chunk size on the tablespace is 8K wasn't sure if doing slightly less than 8K might prevent some overflow problems (it doesn't).

then I tested inserting the same amount of data into a table of line#'s and varchar2(4000) (like user_source) with a hefty initial extent and got:

```
no
indexes  8.63  7.49  7.75

w/pk
index   14.72  15.41  15.32
```

the inserts without indexes is slightly more performant, but we'd clearly have at least a primary key on the table, so the trials with a pk index are indicative of the performance we'd really see.

Also, to delete one row that contains a 800K clob takes a fraction of a second, whereas to delete 10000 80 byte rows (as in the table of vc2 model) takes about 5 to 10 seconds.

Conclusion:

The clobs do seem to be the best data structure for what we are trying to do (emulate a file system). We just need to pay attention to how we size tablespaces and tables, and our programs that read and write to the clobs need to implement a buffering scheme (which is what an OS file system does as well, so it all makes sense).

Creating a text file

- Simple Open/Write/Close paradigm

`ex_ls_api_write.sql`

openFile call

```
function openFile(pFileName in ptn_filename%type,  
                 pMode in ptn_mode%type, -- R W A  
                 pShareMode in ptn_mode%type default null) -- S or X  
return ptn_handle%type;
```

- Reading a File `ex_ls_api_read.sql`

LS_API locking modes

- S or Share mode uses autonomous commits to create the file
 - Allows producer/consumer interaction
`ex_ls_api_producer.sql` and `ex_ls_api_consumer.sql`
- X or Exclusive mode uses normal Commit/Rollback processing `ex_ls_api_x_lock.sql`

Getting directory contents

- Using the LSFileObj

- LIST function allows filtering on name

```
member function list(p_file_filter in varchar2 default '%',  
                    p_sort_column in varchar2 default 'FILE_INAME',  
                    p_sort_type   in varchar2 default 'ASC',  
                    p_owner_filter in varchar2 default '%')  
  
return ls_fileobj_list
```

- Full API for other file manipulation

ex_ls_fileobj.sql

Query File Hierarchy

- Using `connect_by` syntax to match file with it's parent directory
- Can walk "up" the path, or "down" by changing the starting point and connect by prior clauses ex_file_hierarchy.sql

```
select level pLevel,file_id, file_name, file_type
from ls_file f
  start with file_id = 0
 connect by prior file_id = file_path_id;
```

LS_API Security

- Based on grants to users or roles like database permissions
- Direct user privs take precedence over role privs
- A privilege query starts at the file level and traverses up the hierarchy until a permission is found that allows or disallows access.
- If no privilege specified then access is not allowed

LS_API Java components

■LSLoader and LSUnloader

■Bulk Load and Unload from local filesystem to LS_API

Parameter	Description	Default
Username/password @tns_name	Required parameter that specifies which oracle account (username, password and tns entry) is used to connect to the database. Specifying the @ is required. If connecting to a local database then tns_name can be left blank. -	None
-s	Source folder. Specifies an absolute path to the source folder for loading or unloading. In the loading case, then the path is a specification to a drive letter and path. For unloading then this specifies an absolute directory path within the LS system.	Default working directory or folder
-t	Target folder. Inverse of the -s parameter	Default working directory or folder
-r	Recursive search. Specify this parameter to have the utility recursively search subdirectories.	False
-o	Overwrite. Specify this parameter to have the utility overwrite existing files of the same name.	False
-c	Continue if errors. Specify this parameter to have the utility continue processing the next file in the source folder after an error occurs. By default the utility will abort on the first error.	False
-f	A file pattern string using the DOS convention of asterisk for wildcards. Note you must enclose this parameter in quotation marks if you specify any wildcards.	"*.*)"

Add Ons

- SQLPlus Interpreter (SPLUS)
 - Layered on top LS_API
 - Input are "script" files. Output is spooled output from script run.
- PDF format can be generated
 - See <http://www.dartmouth.edu/~gmenchen/>
- `ex_splus.sql`

Where to go from here

- Configuring a DAD correctly could allow content to be served directly by HTTP_SERVER as if it were a web server
- Create a WebDAV interface
- Contributions to improving the source can be made on sourceforge.net