Processing Large Character Objects

Part 1

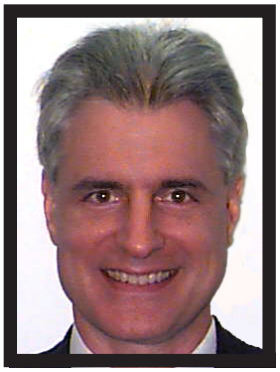
An Oracle "How To" Paper

2007

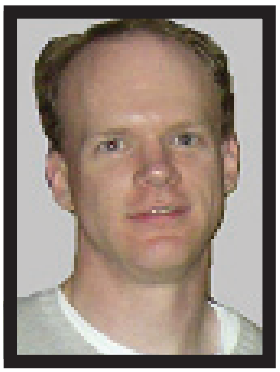
Presented By:

Bob Taylor

William Indest



Jeremy Blankenship



ORACLE | SIEBEL

All generalizations are false, including this one.

Mark Twain

Why Use Large Objects?

A **B**inary **L**arge **OB**ject, also known as a blob, is a collection of binary data stored as a single entity in a database management system. Blobs are typically images, audio or other multimedia objects, though sometimes binary executable code is stored as a blob. Database support for blobs is not universal.

Blobs were originally just amorphous chunks of data invented by Jim Starkey at DEC, who describes them as "the thing that ate Cincinnati, Cleveland, or whatever"[1]. Later, Terry McKiever, a marketing person for Apollo felt that it needed to be an acronym and invented the backronym Basic Large Object. Then Informix invented an alternative backronym, Binary Large Object. Today many people believe that blob was originally intended as an acronym for something[2].

The generic term for these objects is LOB or Large Object. LOBs are a set of datatypes that are designed to hold large amounts of data. A LOB can hold up to a maximum size ranging from 8 terabytes to 128 terabytes depending on how your database is configured. Storing data in LOBs enables you to access and manipulate the data efficiently in an application[3].

This section introduces different types of data that you may encounter in the wide wonderful world of tech and discusses which kinds of data are suitable for large objects. In general, all applications must deal with the following kinds of data:

Simple structured data: These data can be organized into simple tables that are structured based on business rules.

Complex structured data: This kind of data is complex in nature and is suited for the object-relational features of the underlying database such as collections, references, and user-defined types.

Semi-structured data: This kind of data has a logical structure that is not typically interpreted by the database. For example, an XML document that is processed by your application or an external service, can be thought of as semi-structured data.

Unstructured data: This kind of data is not broken down into smaller logical structures and is not typically interpreted by the database or your application. A photographic image stored as a binary file is an example of unstructured data.

Large objects are suitable for these last two kinds of data: semi-structured data and unstructured data. Large objects features allow you to store these kinds of data in the database as well as in operating system files that are accessed from the database.

Overview of BLOBs

---

1. Starkey, James. The true story of BLOBs
2. Wikipedia definition of Binary Large OBject.

3. Reference: Oracle® Database Application Developer's Guide - Large Objects 10g Release 2 (10.2) Part Number B14249-01

The following table lists the large object datatypes and provides a brief description of each:

Large Object Datatypes	Description
BLOB: Binary Large Object	Stores any kind of data in binary format. Typically used for multimedia data such as images, audio, and video.
CLOB: Character Large Object	Stores string data in the database character set format. Used for large strings or documents that use the database character set exclusively. Characters in the database character set are in a fixed width format.
NCLOB: National Character Set Large Object	Stores string data in National Character Set format. Used for large strings or documents in the National Character Set. Supports characters of varying width format.
BFILE: External Binary File	A binary file stored outside of the database in the host operating system file system, but accessible from database tables. BFILEs can be accessed from your application on a read-only basis. Use BFILEs to store static data, such as image data, that does not need to be manipulated in applications. Any kind of data, that is, any operating system file, can be stored in a BFILE. For example, you can store character data in a BFILE and then load the BFILE data into a CLOB specifying the character set upon loading.

Description

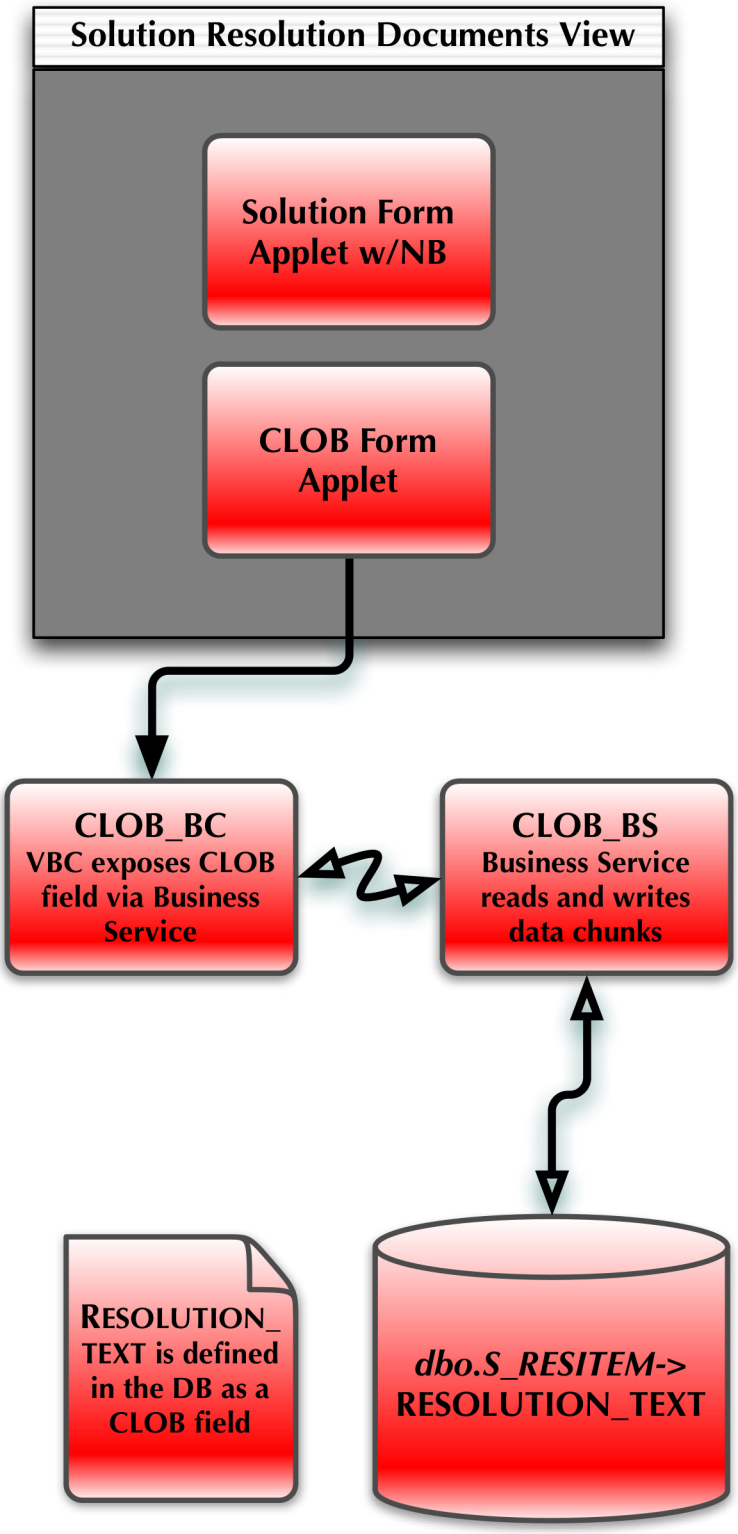
Today Siebel supports the CLOB data type. However, these very large data types are difficult to display and the effectiveness of using a text area of this size would prove ineffective both in terms of usability and system/network performance.

The CLOB Business Service and VBC supports large text objects, it…

- Reads a CLOB field in chunks of specified size – 25KB default (Configurable in the eScript)
- Stores each chunk as a row of data in a VBC
- Allows the navigation and update of these data using a standard form applet to page through the data.

The figure shown below shows the solution topology and should help you understand the basic components of this solution.

Note: The business service supplied uses a database CLOB field. You could use the same code to support large text files or any large data store by changing the appropriate eScript for Query and Update.



Note: This solution demonstrates the basics on how to display and edit large text objects using native Siebel components. In a production implementation where many large objects are accessed in parallel, memory usage would need to be considered and the solution modified as necessary.

Instructions

Eggheads unite! You have nothing to lose but your yolks.

Adlai Stevenson

Overview

First you will create a CLOB. Next, you will import an sif file. After the import, you will have the following objects: a Business Service, a Virtual Business Component and a Form Applet. You will use the Solution Business Object and modify a view with the imported applet to display the CLOB Virtual Business Component. You will change some server and database parameters in the business service to reflect your particular situation. The parameters are highlighted in the doQuery and doUpdate code snippets in the Code section. You will make some minor configuration changes - create a link and update a Business Object.

Create a CLOB

You will convert the existing field, RESOLUTION_TEXT, in the table, S_RESITEM in the demo SQL database, OLTP_SIA. Typically you would not implement CLOB's in the Siebel database directly, we do so here for simplicity. By the way, you can change the eScript to reference other databases, tables, fields as needed.

Let's change the above table field to become a CLOB. To do so follow these steps:

- Launch Enterprise Manager for MS SQL Server
- Open the 'OLTP_SIA' database
- Click on the Tables icon
- Right Click on table S_RESITEM
- Select 'Design Table' from popup menu
- Select field RESOLUTION_TEXT
- Change data type from ntext to text (don't worry about the length)
- Save your changes, read the warning, click "Yes" and exit.

Congratulations! You have just created a CLOB field. MSSQL Server will update this field in place saving all the existing data; existing Siebel references to this field will continue to work and treat it as a varchar data type of the appropriate length.

Import the SIF File

Open Siebel Tools and import the 01_CLOB_Solution.sif file. For conflict resolution, use "Merge the object definition from the archive file with the definition in the repository". The import will load an applet, CLOB_BC Form Applet, a virtual business component, CLOB_BC, and a business service, CLOB_BS. Go to the Project list and lock two projects: the 01_CLOB_Solution project and the Solution project. Later you will add the CLOB_BC to the Solution Business Object but before you do that you first

need to create a simple link. You will create the link in the Solution project so while you are in the Project list, lock the Solution project. This step saves you from having to return to the Project List.

Create a new Link

Go to the Link List and create a new link with the following properties:

Name: Solution/CLOB_BC

Project: Solution

Parent Business Component: Solution

Child Business Component: CLOB_BC

Source Field: Id

Destination Field: Id

Add the VBC to the Business Object

Go to the Solution Business Object and add the CLOB_BC business component to the object. Use the Solution/CLOB_BC link.

Add the Applet to the View

Prior to navigating to the Solution Resolution Documents View, compile the CLOB_BC Form Applet. Then invoke Edit Web Layout on the Solution Resolution Documents View. In Base Mode, delete the lower applet called SR Resolution Item List Fram Read Only, and add the CLOB_BC Form Applet underneath the Solution Form Applet w/NB. Under View Web Templates, open the View Web Templates Item and change the Applet mode for the CLOB_BC Form Applet to Edit mode.

Specify Server and Database parameters

Go to the CLOB _BS business service and invoke Edit Server Script on the selected record. Refer to the code topology diagram for the overview as well as the code section. For the two functions, doQuery and doUpdate, find the following in each:

```
//---------------------------------------------------------------------
// Create a connection to the database and open the table in read mode
// The strCn needs to have the name of the server where the database
// resides; the Database variable should be the name of the database
// to which you connect; this is the same name that you access with
// the Enterprise Manager for MS Sql Server.
cn = COMCreateObject("ADODB.Connection");
strCn = "Server=WP7014;Database=OLTP_SIA;UID=SADMIN;PWD=SADMIN;";
```

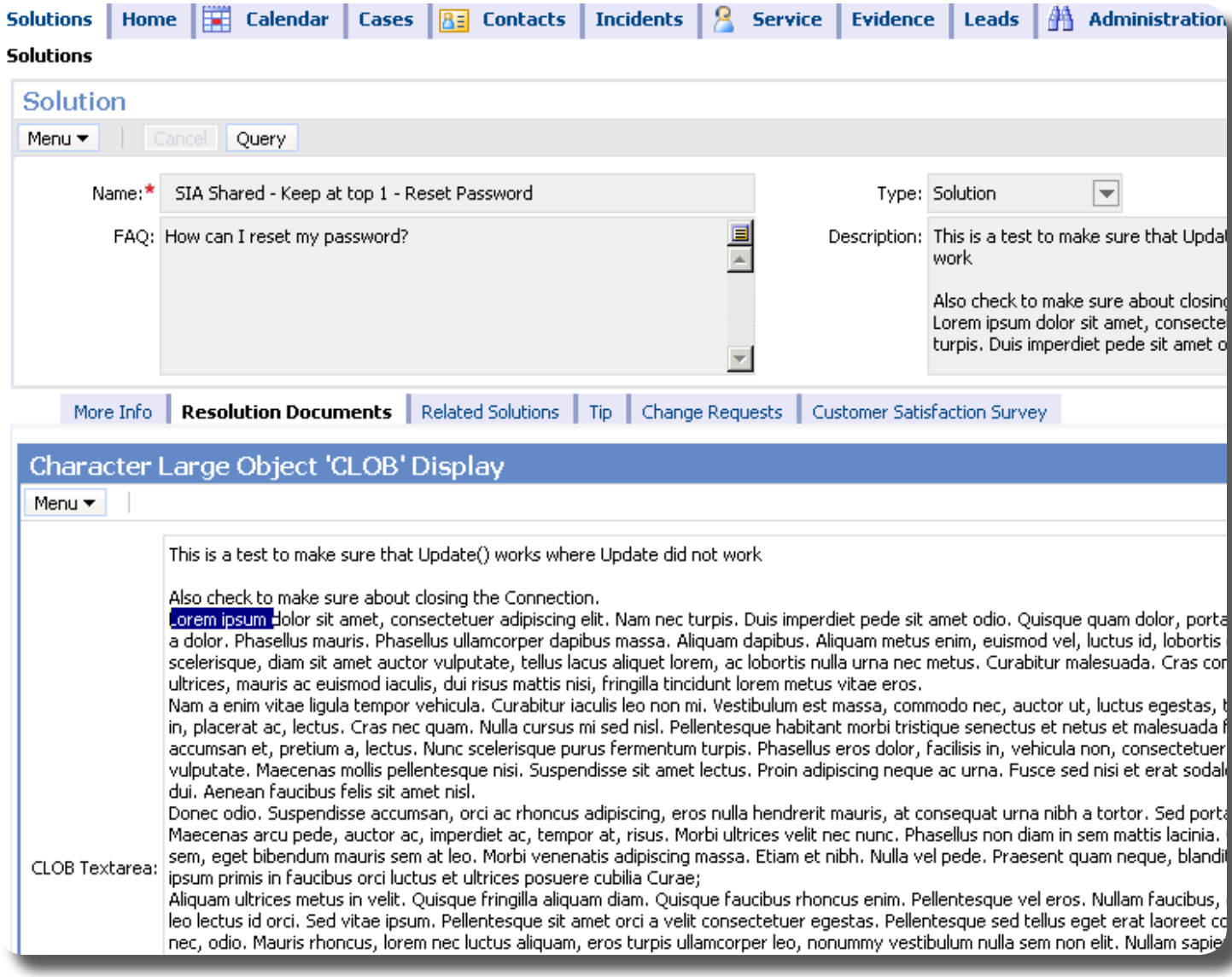
Change the "Server=" parameter to COMPUTERNAME of your environ-

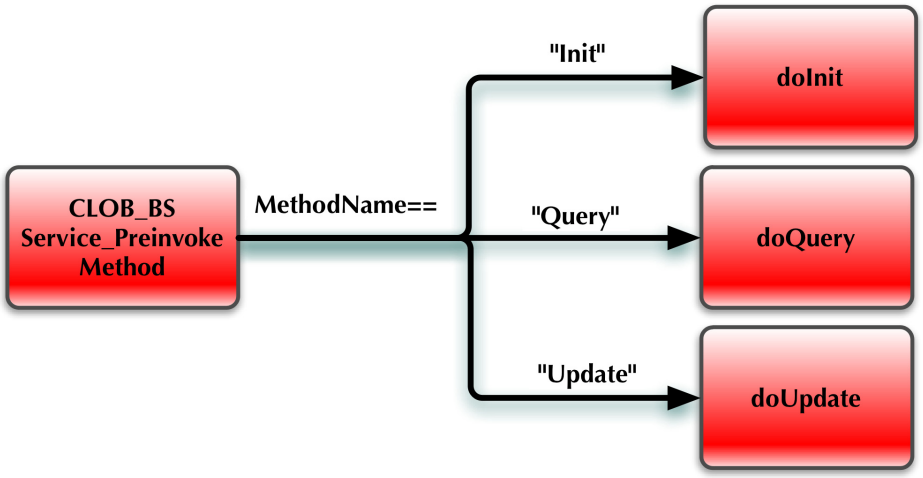
ment. You can find your computer name in several ways: 1) Right mouse click on "My Computer" and select properties then click on the Computer Name tab; 2) Invoke the Run command from the Start menu and enter "echo %COMPUTERNAME%". Do not include the quotes; hit run and the variable's value will be displayed in a DOS window. This value should replace WP7014 (after "Server="). Check that the database name is the same as the database in which you created a CLOB (Step 1).

Note: there are two places you need to change parameters.

Compile and Launch

Compile the projects 01_CLOB_Solution and Solution, and start the application. Navigate to Solutions -> All Solutions view and then drill down on any record, you should see the view below. You can paste in large amounts of data and either explicitly save using the applet menu or navigate to the next record.





The above figure depicts what the CLOB_BS business service does at a high level and delineates the three invocations, "doInit", "doQuery" and "doUpdate". Here is the business service code which manipulates the CLOB.

Ah, code...

```
function Service_PreInvokeMethod (MethodName, Inputs, Outputs)
{
if (MethodName == "Init") {
        doInit( Outputs );
        return(CancelOperation);
}
if (MethodName == "Query") {
        doQuery( Inputs, Outputs );
        return(CancelOperation);
        return;
}
if (MethodName == "Update") {
        doUpdate( Inputs, Outputs );
        return(CancelOperation);
        return;
}

return(ContinueOperation);
}
function doInit( Outputs ) {
        Outputs.SetProperty("Page","");
        Outputs.SetProperty("Page Number","");
        Outputs.SetProperty("Id","");
        return;
}
```

```
function doQuery(Inputs , Outputs) {
TheApplication().Clobdata = new Array();
TheApplication().Clobdata[0] = "";
TheApplication().Cloblast = 0;
var str = "";
var subPS;
var cn = new Object;
var strCn = "";
var rset = new Object;
var tmpStr = "";
var offset = 0;
var actSize = 0;
var fld;
var i = 0;
var propset;
var propname;
var Chunksize;

var Id = "";

//---------------------------------------------------------------------
// Get input row id of record to process and store in global variable

propset = Inputs.GetChild(0);
propname = propset.GetFirstProperty();
Id = propset.GetProperty( propname );
TheApplication().ROW_ID = Id;

TheApplication().Cloblast = 0;

//---------------------------------------------------------------------
// Create a connection to the database and open the table in read mode
// The strCn needs to have the name of the server where the database
// resides; the Database variable should be the name of the database
// to which you connect; this is the same name that you access with
// the Enterprise Manager for MS Sql Server.

cn = COMCreateObject("ADODB.Connection");
strCn = "Server=WP7014;Database=OLTP_SIA;UID=SADMIN;PWD=SADMIN;";
```

```
cn.Provider = "sqloledb";
cn.Open( strCn );

rset = COMCreateObject("ADODB.Recordset");
rset.Open("S_RESITEM",cn,0,3);

//---------------------------------------------------------------------
// Select the correct row to update, this is the row_id originally passed
// in the doQuery function from the link
// e.g rset.Filter = "ROW_ID = '1-I87B'";

Clib.sprintf(str,"ROW_ID = '%s'",Id);
rset.Filter = str;
fld = rset.Fields("RESOLUTION_TEXT");

//---------------------------------------------------------------------
// Read the CLOB data into both a global variable array for later use
// in update processing and the property set used by the VBC

i = 0;
offset= 0;
Chunksize = 25000;// size of chunk to display at one time in a page
actSize = fld.actualSize;
while(offset <= actSize) {
        TheApplication().Clobdata[i] = fld.GetChunk(Chunksize);
        TheApplication().Cloblast = i;

        subPS = TheApplication().NewPropertySet();
        subPS.SetProperty( "Page", TheApplication().Clobdata[i]);
        subPS.SetProperty( "Page Number", i );
        subPS.SetProperty( "Id", Id );
        Outputs.AddChild( subPS );
        offset = offset + Chunksize;
        i++;
}
cn.Close();
return;
}
```

```
function doUpdate(Inputs , Outputs) {

var str = "";
var subPS;
var cn = new Object;
var strCn = "";
var rset = new Object;
var fld;
var i =0;
var propname;
var pagenum;
var fieldname;
var strChunk;



//---------------------------------------------------------------------
// Create connection to the database and open the table in edit mode.
// The strCn needs to have the name of the server where the database
// resides; the Database variable should be the name of the database
// to which you connect; this is the same name that you access with
// the Enterprise Manager for MS Sql Server.

cn = COMCreateObject("ADODB.Connection");
strCn = "Server=WP7014;Database=OLTP_SIA;UID=SADMIN;PWD=SADMIN;";
cn.Provider = "sqloledb";
cn.Open( strCn );
rset = COMCreateObject("ADODB.Recordset");
rset.Open("S_RESITEM",cn,2,3);
//---------------------------------------------------------------------
// Select the correct row to update, this is the row_id originally passed
// in the doQuery function from the link
// e.g rset.Filter = "ROW_ID = '1-I87B'";

Clib.sprintf(str,"ROW_ID = '%s'",TheApplication().ROW_ID);
rset.Filter = str;
fld = rset.Fields("RESOLUTION_TEXT");

fld.Value = "";
```

```
//---------------------------------------------------------------------
// Loop through the input property set and find the Page number of the
// record that has been updated
//
for( i=0; i<Inputs.GetChildCount(); i++ ) {
        subPS = Inputs.GetChild(i);
        propname = subPS.GetFirstProperty();
        while ( propname != "" ) {
                fieldname = subPS.GetProperty("Field Name");
                if ( fieldname == "Page Number") {
                        pagenum = subPS.GetProperty("Field Value");
                }
                if ( fieldname == "Page" ) {
                        strChunk = subPS.GetProperty("Field Value");
                }
                propname = subPS.GetNextProperty();
        }
}

for( i=0; i <= TheApplication().Cloblast; i++ ) {
        if (i == pagenum) {
                fld.AppendChunk( strChunk );
        } // new one
        else {
                fld.AppendChunk( TheApplication().Clobdata[i] ); // else original one
        }
}
rset.Update();
cn.Close();
return;
}
```

Beware of the bugs in the above code; I have only proved it correct, not tried it.

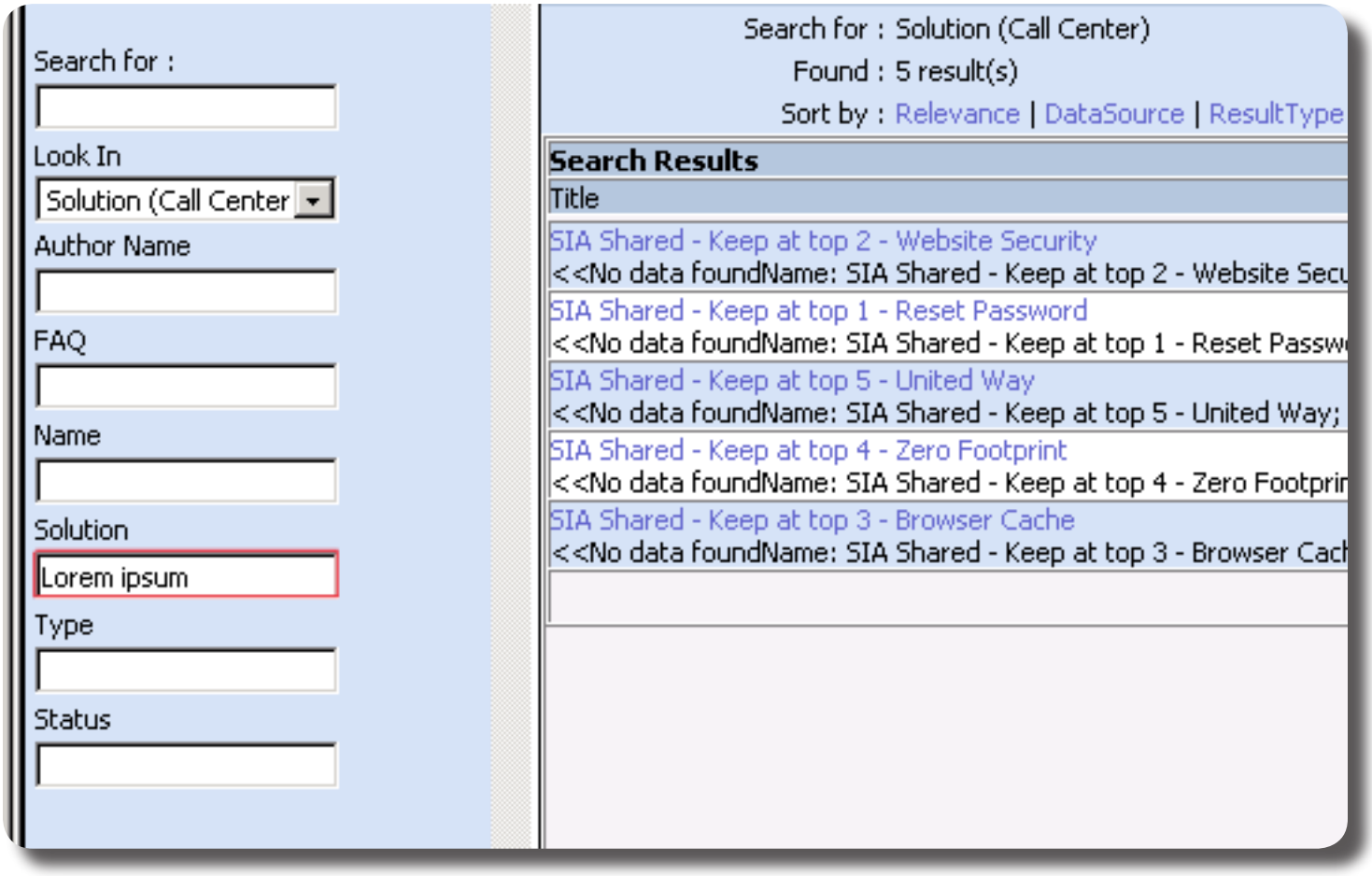
Donald Knuth

Computers are useless. They can only give you answers.

Pablo Picasso

Postscript

You may be wondering if these BLOBs are searchable. Check the following screenshot:



Lorem Ipsum stuff

Lorem ipsum content was added to each CLOB field of five Solution records and as you can see, the content (Lorem ipsum) in the CLOBs is indeed searchable.

Note: Lorem Ipsum is dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC however there are many variants and randomizations. The basic driver for Lorem Ipsum is a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. For more info, google it. When you return home you may tell your significant other that you were reading Cicero today; let the sparks fly!

It's been a blast from the past

Historical Footnote

Here is a list of the types of modifications that we needed to add to the original CLOB.sif file which was created in July, 2002. Most of the 114 modifications were on the fields of the Solution Business Component.

It is remarkable that after five years the stability of the interface remains so solid.

1. UI_FREEZE="N"
2. UPGRADE_BEHAVIOR="Preserve"
3. APPLICATION_CODE="STD"
4. MESSAGE_DISPLAY_MODE="UserMsgwithErrorCodeOnly"
5. TYPE="Non-Transient"
6. DISABLE_SEARCH="N"
7. DISABLE_SORT="N"
8. STATE_MANAGEMENT_TYPE="Stateful"
9. WEB_SERVICE_ENABLED="N"
10. ADD_TO_HISTORY="Y"
11. DISABLE_PDQ="N"

Title: Processing Large Character Objects
Authors: Bob Taylor, William Indest and Jeremy Blankenship
Managers: Eileen Smyth, James Perry and Todd Kyle

Oracle Corporation
World Headquarters
500 Oracle Parkway
Redwood Shores, CA 94065
U.S.A.

Worldwide Inquiries:
Phone: +1.650.506.7000
Fax: +1.650.506.7200
oracle.com

EOF