Secure Oracle Web Access

Bill Cattey, Diane Delgado, Bruce Lewis, Miki Lusztig, Steve Neiterman

December 4, 1996

Contents

1 Introduction

There is currently a great deal of interest at MIT in accessing information in Oracle databases from the World-Wide-Web. The Web technology is very appealing because of its ability to create simple, system independent GUI clients; thus, dispensing with the need to build, test, and maintain the interface on each type of client machine.

This document discusses issues relating to secure Web access to Oracle databases and proposes methods by which this can be accomplished. The work discussed here is based on prototype work which was performed to investigate using the Oracle database for the Practice-Discovery project database.

The work discussed here was implemented on Solaris 2.4; we do not address other (non-unix) operating systems in this document.

2 Limitations of Web Access

Though a Web Client interface is simple to develop, maintain, and deploy, it is not a complete replacement for technologies such as Powerbuilder or Oracle's SQLNet, both of which work together to provide remote access client interfaces.

Web database access is suited for the following types of situations:

- The database will be accessed infrequently for writes.
- The database is expected to have a small to moderate number of read requests.
- There is a limit imposed by Oracle on the amount of web data which can by dynamically generated so queries which would return extremely large amounts of data are not suited for web access.
- It is not suited for random sql queries, e.g., the type of query which can be performed using *sqlplus*.

Web access to databases is not suited at all for high-volume transaction processing.

3 History

The initial prototype for Oracle access using the Practice Discovery database used Oracle Web server and tools provided with the Oracle 7.3 release. The architectural model for this scheme is depicted in the drawing labeled *Web Access architecture using Oracle components.*

The Oracle Web Server, web agent and the stored pl/sql procedures reside on a database server machine and the web client accesses the database procedures remotely via the Oracle web server. The Oracle Web Agent is a CGI-compliant program which is spawned by the Oracle web sever to execute the stored pl/sql procedure requested by the client.

The methods of security supplied by the Oracle web server in Oracle 7.3 are basic and digest authentication. Neither authentication method meets the security needs of the MIT environment, which is largely based on kerberos authentication.

The initial security infrastructure mandated an MIT slogin service which required that CGI programs invoke slogin library routines to retrieve the Web client's kerberos identity. Because we did not have access to the sources for the Oracle Web Agent, we created a replacement CGI program

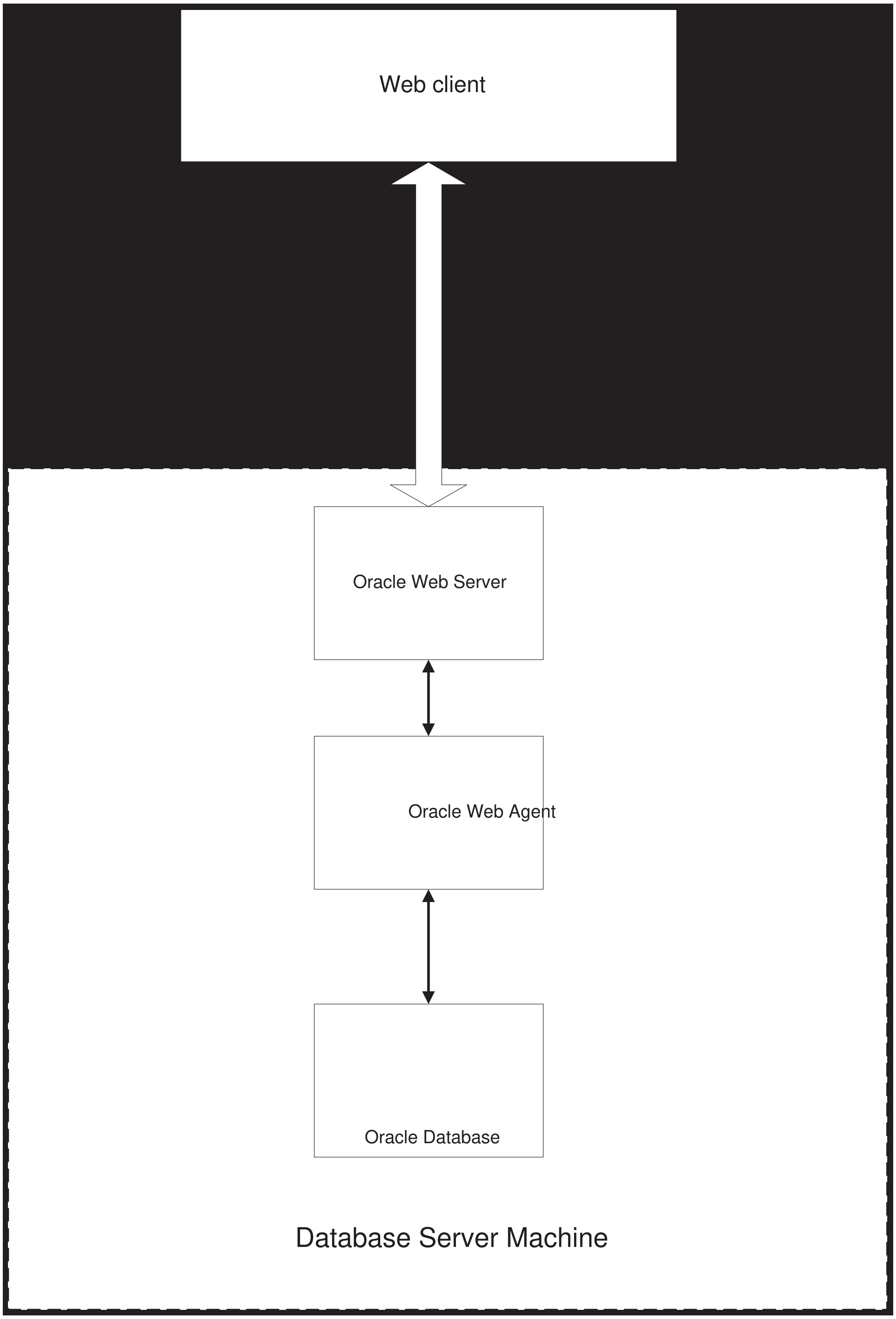

Figure 1: Web Access architecture using Oracle components

for the web agent using a mix of embedded SQL and OCI calls which could later be integrated with the slogin security method.

At this writing, the MIT's infrastructure for secure web access is now based on the use of SSL (Secure Sockets Layer) in conjunction with x.509 certificates. Under this scheme, MIT will implement its own Certificate Authority which will manage the granting and verification of x.509 certificates. A web client will request certificates from this authority, and present them upon request to an SSL-compliant Web Server (e.g., Apache). The Web Server will acquire a certificate for the MIT CA and use both the MIT CA certificate and the client's certificate to verify the identity of the Web client. As a side effect of this verification process, the web server will derive the client's distinguished name which then can be used for access control checks.

The x.509 specification allows flexibility in the format of the x.500 distinguished name which is part of the certificate. Because the current security infrastructure at MIT is kerberos based, MIT's distinguished name format will include kerberos principal information. (The Kerberos prinicpal name is contained in the *Email* component of the distinguished name.) CGI programs can use this kerberos principal information to perform access checks in lieu of the distinguished name if they so desire. The actual format of the MIT distinguished name has not been finalized at this time.

The Oracle Web Server V1 which is shipped with Oracle 7.3 does not support SSL or x.509 certificates. Oracle Web Server 2.0 purportedly does; however, it is unknown whether its distinguished name format will be compatible with the MIT distinguished name format.

4 Architectural Overview

Because of the requirement that the Web security scheme support MIT's x.509 certificates, we chose to replace the Oracle-supplied components in the initial prototype with public-domain or internally developed components. The current architecture is depicted in the drawing labeled *Current Architecture for Oracle Web Access.*

Under this implementation, the Apache Web Server, db.cgi program, and the Oracle database reside on the same host.

Note that if in the future, the Oracle Web products support the distinguished name format used by MIT, then we can replace the Apache Web Server and the db.cgi program with their Oracle counterparts.

4.1 Apache Web Server

The Apache Web Server is the most widely used public-domain web server. There is an SSL source overlay with x.509 certificates available for this web server. Thus, the sources can be modified, if necessary, to accommodate any changes required to integrate with the MIT certificates.

4.2 DB.CGI

Db.cgi is the cgi-compliant embedded SQL C program previously mentioned which manages execution of stored PL/SQL procedures by web clients. This code was internally developed at MIT and can be modified to accommodate MIT distinguished names.

The db.cgi program supports the ability to execute *any* authorized stored procedure in a database. Db.cgi is application independent, and does not need to be modified when used with different Oracle databases or different Oracle stored procedure packages.

The db.cgi program performs the following functions:

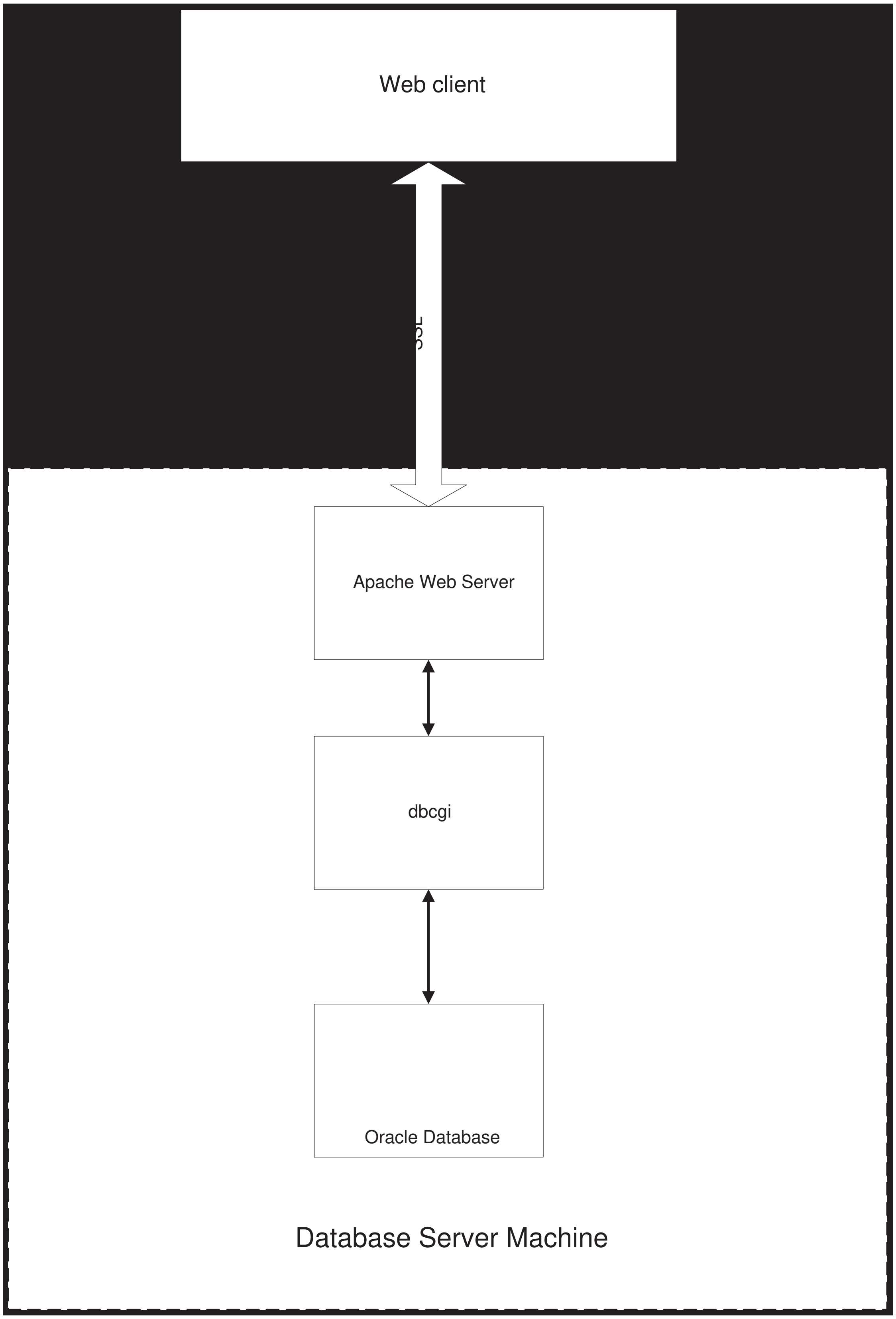

Figure 2: Current Architecture for Oracle Web Access

- possibly perform some translation of the distinguished name
- setup the proper execution environment for the stored procedure
- connect to the database
- fetch and parse the cgi input according to whether GET or POST method is indicated.
- generate, parse and bind an sql statement to invoke the stored procedure
- execute the stored procedure using the parsed sql statement.
- fetch the html text generated by the stored procedure and write it to stdout.

The db.cgi program will be discussed in more detail in later sections.

4.3 Oracle Web-Related Packages and Stored Procedures

As part of its WWW application development tools package, Oracle provides several database pl/sql packages to facilitate dynamic generation of Web pages. These packages are located in the Oracle product install tree under *ORACLE_HOME/product/7.3.2/ows/admin*. These packages must be loaded into the database in order to generate dynamic Web pages using data from the Oracle database. (These packages are also used in conjunction with the Oracle Web Server V1 and the Oracle Web Agent.)

The following packages are contained in the Oracle WWW application development package:

- owa - utility routines to allow cgi programs to retrieve dynamically generated web pages and to place CGI environment variables in the stored procedures execution environment.
- owa_util - more utility routines to perform actions such as print all environment variables passed from the cgi program, output a redirect url, etc.
- htp - generates htp formatting directives. These are actually analogous to writing macros which actually expand into htp directives. This package implements a procedural version.
- htf - function version of *htp* procedures.

With these packages, a programmer would create his own stored packages and procedures using pl/sql to access the information in the database tables, and call the oracle-supplied web-related package routines to format the data into html web pages. As part of these packages, Oracle provides procedures which allow the db.cgi (or other cgi) program to pass CGI environment information to the application stored procedures.

The following pl/sql procedure prints project name and description for all projects which are potential or active for the specified practice area. The html data is displayed so that the user can click on the project name to get more detailed information on that project. (The latter action has the effect of the client's sending a request to invoke the *project_by_number* stored procedure to lookup the project's complete information.)

```
PROCEDURE gen_proj_links (pract char) IS
    pname project.name%TYPE;
    pnum  project.projnum%TYPE;
```

```
        pdesc project.description%TYPE;
        cbuf varchar2(255);
        CURSOR get_proj(pr char) IS  select name, description, projnum
               from project WHERE practice = pr and
               (status = 'potential' or status = 'in process');
    BEGIN
        OPEN get_proj(pract);
        LOOP
            FETCH get_proj INTO pname, pdesc, pnum;
            EXIT WHEN get_proj%NOTFOUND;
            cbuf := htf.anchor(
               'pdproj.project_record_bynumber?proj_num='||to_char(pnum),
                pname, NULL, 'HREF');
            htp.print(cbuf);
            htp.print(pdesc);
            htp.para;

        END LOOP;
        CLOSE get_proj;
    END gen_proj_links;
```

The stored procedure is invoked from a web client by specifying a URL with the following format: https://$< server - name >$/$< oracle\_sid >$/db.cgi/$< stored\_procedure + arguments >$

The stored procedure argument and argument values are passed using the standard CGI GET or POST methods. The $< oracle\_sid >$ indicates to db.cgi which oracle database to use for the connection.

When actually executed, this stored procedure will generate the following

```
<A HREF="pdproj.project_record_bynumber?proj_num=1" HREF>
   UROP Database Upgrade</A>
There is currently a database of UROP students which feeds
information to the Student Information System.  Early analysis
is that this db should be upgraded and integrated better into the
student information environment.
<P>

<A HREF="pdproj.project_record_bynumber?proj_num=2" HREF>
   Distributed Library Initiative</A>
Working space shared jointly by the Libraries and Information
Systems.  A variety of programs and projects fit in this effort to
improve electronic library services.
<P>
```

The generated html data is retrieved from the owa package by db.cgi and subsequently delivered to the web client for display.

5 Client/Server Security

It is not clear at this time whether the apache web server will extract the client's kerberos identity from the distinguished name or whether the db.cgi program will perform this. It is also possible that the full distinguished name will be passed through untranslated to the database stored procedures.

We recommend that whichever entity is responsible for managing the client identity, that the type of information extracted be configurable.

The client's identity (whether it be the full distinguished name or the kerberos principal) is contained in the REMOTE_USER CGI environment variable.

6 Database Security

Typical database security involves creating database user accounts (within the database) and granting access privileges or roles to these accounts. Database accounts are protected either by password or by having the database use the user's operating system user id.

Some database applications embed access control into the application in lieu of or in addition to individual database user accounts.

The x.509 security does not allow the typical database security methods based on individual user database accounts to be employed because we have access only to the user's login name and not his password which would be needed to perform an actual login to the database. It does, however easily lend itself to the method in which the access control is embedded into the application.

Because no one access control method is acceptable for all applications, we recommend that db.cgi support application implemented access control via the REMOTE_USER or SSL_CLIENT_DN information and database access control by connecting as the oracle user specified REMOTE_USER or SSL_CLIENT_DN.

6.1 Application Access Control

Under this method the dba would create a web user account in the Oracle database and grant it appropriate privileges. The db.cgi program will connect to the database using this account. The username and password of the web user account will be kept secured in a file on the local disk. This file will be read by db.cgi which it is invoked and the information contained therein will be used as the connect string for the *EXEC SQL CONNECT*. The db.cgi will pass the REMOTE_USER or SSL_CLIENT_DN information to the pl/sql procedures so that it may be used by the application, if desired, to perform additional access control checks.

This type of access control method is suitable for read-only web access to an Oracle database.

6.2 Individual Sign On

Under this method, db.cgi will connect as distinct database users; the user account used by db.cgi will be determined by the web client's identity which was derived from the x.509 certificate.

There are a few issues regarding the use of distinct client identities:

- The db.cgi program needs access to database usernames and passwords in order to connect as a specific database user.

- Some databases may require both Web access and SQLNet access. Thus, it is desirable that the individual sign on scheme does not preclude database access using SQLNet or other third-party products.

- Because of security concerns, it is preferable to not maintain database account names and passwords in cleartext on local disk.

The proposed solution uses two database accounts for each database user who requires access to Oracle via the web. The first account is the tradition type of database account which is used to login to Oracle via SQLNet or sqlplus. Note that there are some cases where users do not have normal database accounts. In such cases the traditional account is not created.

The second account is used exclusively by db.cgi for connect purposes. The web account name would have the form < *user* >*_web*. The account would be initially created by the DBA and its password maintained separately by the db.cgi program. The web accounts will need to be granted the same roles and privileges as their traditional account counterparts.

The db.cgi is responsible for generating a random password for the web user database accounts, connecting to oracle, and changing the password for the web account. The db.cgi program will then be able to log into Oracle as *user_web*.

When db.cgi changes the web account passwords, it connects as *oramaint*, which is an account created by the DBA with appropriate privileges to alter user passwords.

When db.cgi is invoked it will perform the following steps to setup the connection to the database:

- derive the client's identity from the REMOTE_USER information. Note that because our architecture specifies the use of SSL with x509 certificates, db.cgi can trust the client identity information which the web server has provided it.

- generate a radom password

- connect to database as *oramaint* and modify the password for the *user_web* account.

- connect to the database as the web user. If no account information was found found for the REMOTE_USER by the previous steps, then connect as *oracleweb*

A utility will be provided to create web database user account and grant it the same roles and privileges as its corresponding traditional user account.

6.3 User and Group Id of DB.CGI

It is recommended that db.cgi run as a non-privileged user, e.g., *oramaint*. All of its datafiles should be protected so that they are not world readable. Because db.cgi has certain database privileges and its database password needs to be protected, we do not recommend the use of the widly-used *nobody* user.

7 Remote Database Access from DB.CGI

The type of configuration we have addressed thus far requires that the Web Server, db.cgi program, and Oracle database reside on the same host. We have not addressed the scenario where the Oracle database resides on a different host from the db.cgi program.

Preliminary discussions on this matter have resulted in the following recommendations:

- Although there is a kerberos protocol adapater available for SQLNet, we feel that requiring the db.cgi program to acquire and maintain kerberos tickets is too complex for this purpose.
- We feel that it is sufficient protection for the dbgci to connect to the Oracle database via SQLNet using an encrypted channel so that the username/password information is not passed in the clear. SQLNet has adapters which can support this type of connection.
- At this time, we don't have any plans investigate this further.

8 DB.CGI Implementation

This section contains low-level information regarding the actual implementation of the db.cgi program. The information contained in this section is intended mostly for programmers who are familiar with the CGI specification, Oracle packages and stored procedures, and with Oracle embedded SQL application development.

Db.Cgi is an embedded SQL C program which also uses OCI (Oracle Call Interface) routines to allow Web clients to invoke stored database procedures. Embedded SQL is a standard method of accessing relational databases from languages such as C, Fortran, and Cobol. It allows the programmer to use SQL syntax from a C program to perform RDBMS queries:

```
EXEC SQL DELETE FROM project WHERE project_name = :pname;
EXEC SQL COMMIT;
```

The above query deletes all records from the *project* table which match the value of *:pname*. Here *pname* is an actual C variable containing a name of the project to be deleted. The *:* indicates to the Embedded SQL precompiler that this is a C *host* variable.

During pre-compilation the Embedded SQL precompiler will translate the *EXEC SQL* calls to appropriate Oracle library calls, set up private data structures to pass the information about the query to the library calls, and bind the values in host variables to actual call parameters. The query is then executed via a separate library call, which in effect accesses the database.

In addition to the use of Embedded SQL, db.cgi also uses OCI routines to effect dynamic parsing an execution of stored procedures requested by web clients. OCI is at a lower level than embedded sql. It is analogous to the programmer's having to perform the work of the Embedded SQL precompiler; i.e., the programmer performs the setup, bind and execute work himself using APIs defined in the Oracle Call Interface.

More information about both of these programming methods can be found in the *Oracle Developer's Guide*.

8.1 Accessing The Dynamically Generated HTML Pages

The Oracle RDBMS provides a number of public packages and procedures of which applications may take advantage. These packages are located in *ORACLE_HOME/product/7.3.2/rdbms/admin* and should be loaded into the database by the DBA.

Both the Oracle-web related packages and the db.cgi program use the *dbms_output* package to store and retrieve the html text which is dynamically generated by the application's stored procedures. The html text is stored in private buffers of fixed length. Oracle imposes a maximum on the amount of data which can be stored; this limit is 1,000,000 lines of 255 bytes each.

The db.cgi program sets up the cgi environment variables for the application, sets up its internal buffer using the *dbms_output.enable*, invokes the requested stored procedure, and then retrieves the generated html page from an internal buffer using *dbms_output.getline*.

8.2 CGI Environment Initialization

The db.cgi program must perform some initializations so that the standard CGI environment variables shared between the Web Server and CGI program are also accessible to the pl/sql routines which will dynamically generate the web pages.

The db.cgi program will invoke the PL/SQL database procedure *owa.init_cgi_env* to initialize the CGI environment variable values which are passed into its environment by the Web Server. This allows the database web packages to have access to the CGI environment. The following is a list of environment variables specified in the CGI specification:

```
SERVER_SOFTWARE
SERVER_NAME
GATEWAY_INTERFACE
SERVER_PROTOCOL
SERVER_PORT
REQUEST_METHOD
PATH_INFO
PATH_TRANSLATED
SCRIPT_NAME
QUERY_STRING
REMOTE_HOST
REMOTE_ADDR
AUTH_TYPE
REMOTE_USER
REMOTE_IDENT
CONTENT_TYPE
CONTENT_LENGTH
```

The db cgi program will performs a *getenv* call for each of these variables and call the *owa.init_cgi_env* database procedure to initialize these variables and their values in the application specific database package which implements the web access.

8.2.1 The SCRIPT_NAME Environment Variable

The *SCRIPT_NAME* environment variable contains the pathname component which precedes the name of the CGI script being executed. Db.cgi interprets the last component of *SCRIPT_NAME* as the database name to which it should connect. (This is the *ORACLE_SID* environment variable.) This allows a single db.cgi program to access different databases on the same host.

8.3 Oracle Environment Variables

The standard oracle environment variables, ORACLE_BASE, ORACLE_HOME, ORACLE_DOC, will be read from a file named *.oraenv* located in the same directory as the db.cgi program. This file will have the following format: $< variable = value >$, where each variable/value pair is on a separate line.

This file will be read by the db.cgi program when it is invoked to setup the proper Oracle Environment.

8.4 Interaction Between OCI and Embedded SQL

The only issue with mixing the Oracle Call Interface routines with embedded SQL is to ensure that the OCI routines and Embedded SQL code use the same database connection for their work.

This involves establishing a database connection using embedded SQL *EXEC SQL CONNECT..* and subsequently linking an OCI login context area with the previously established connection via the *sqllda* call.

8.5 Binding and Stored Procedure Invocation

In order to invoke the requested stored procedure, the db.cgi program must create a template sql statement of the procedure and its arguments so that the statement can be parsed and its parameters bound to actual host values.

Typically an application can invoke stored procedures using embedded SQL as follows:

```
EXEC SQL EXECUTE
BEGIN
    dbms_output.enable(:maxlines);
END;
END-EXEC;
```

This works well when the programmer knows at the time the application is being written which stored procedures will be invoked. The PROC compiler will automatically generate the code which will invoke Oracle library routines to parse and execute the embedded sql statement.

In the case of db.cgi, the name of the stored procedure and its parameters is dynamic in that it is specified at runtime by the Web client. Therefore we must dynamically generate an SQL statement to invoke the procedure with its proper arguments during the execution of db.cgi.

The db.cgi program generates a template procedure invocation using the stored procedure's declaration (procedure name, parameter names and types). This declaration is stored in the Oracle RDBMS data dictionary and can be accessed using the OCI *odessp* call.

The template has the form:

```
BEGIN <pname>(:1, :2, ... :n); END;
```

Where there is a one-to-one correspondence between the number and position of the parameters listed in the procedure declaration and the $:1, :2, :n$ parameters.

Next the template statement is parsed using the OCI routine *oparse*. After successful parsing, the actual values of the parameters specified by the Web client are bound to the positional parameters $:1, :2, :n$; using *obndrv*. The statement is now ready for execution using *oexec*.

8.6 DB.CGI Algorithm

The db.cgi program will be implemented using both embedded SQL for C and the OCI (Oracle Call Interface) routines. We use embedded SQL whenever possible because it is easier to implement and understand. However, we need to use OCI routines to access the data dictionary to retrieve definitions of stored procedures.

The db.cgi program performs the following actions:

- connect to the database
- setup OCI login context to use the connection created by the embedded SQL login sequence.
- perform getenv to retrieve the CGI environment variable values.
- call the stored database procedure owa.init_cgi_env to initialize the environment variable values for the owa package.
- call dbms_output.enable to initialize a buffer to store the web page which will be generated.
- call the OCI odessp to retrieve information about the stored procedure which the client has requested.
- if the procedure does not exist, generate an error page and exit.
- call the OCI routine oopen to create a context area for parsing the stored procedure call statement. If there is an error, print error page and exit.
- generate an SQL statement of the form *BEGIN procname(:1, :2... :n); END;* Here *procname* is replaced with the name of the procedure which the client has requested. The parameters *:1, :2...* represent the parameters which are defined for the stored procedure.
- call the OCI oparse routine for the statement we have just generated.
- for each parameter which the client has specified, call obndrn to bind the actual value to one of the parameters listed in the generated statement. IF any errors are encountered we generate an error page and exit.
- call the OCI routine oexec to execute the statement.
- call the stored procedure owa_util.showpage to dump the generate web page into the output buffer created by the *dbms_output.enable* call.
- fetch the data from the output buffer and display it on *stdout*.

8.7 Handling Escaped Sequences and Special Characters

The db.cgi program will need to perform the following translations of input contained in the *QUERY_STRING* environment variable or stdin:

- Translate '+' to ' ';

All other encoded 2 digit hex sequences will be converted from their hex string representations to their corresponding ascii numeric values.

8.8 Overloading

Oracle PL/SQL supports overloading of procedures. With an overloaded procedure, it is impossible for the db.cgi program to determine which definition of an overloaded procedure to invoke. Therefore db.cgi will return an error if it encounters an overloaded procedure.

9 Outstanding Issues

Outstanding issues related to this work are:

- *fluctuating MIT distinguished name format* - Because the MIT distinguished name format could change for some period of time, it is recommended that a callable library routine be provided to extract the kerberos prinicipal information from the distinguished name. This will insulate web servers and/or cgi applications from the changes in the distinguished name format.

  Jeff Schiller will supply patches for the Apache server and appropriate kerberos-id related routines.

- Threshold - We need to characterize the performance of and determine what the performance limitations are to database Web access.

- Remote db access from dbcgi - We need to determine whether there is sufficient need for this to warrant further investigation.

10 Oracle Configuration

This section describes how to configure ORACLE for use with db.cgi.

10.1 Install The Oracle Web Packages

The Oracle-web packages must be installed in the Oracle database, since db.cgi relies on the functionality provided by these packages.

To install the oracle-web packages execute the following pl/sql script: *product/7.3.2/ows/admin/owains.sql*. This will install all of the required WWW packages and procedures.

After installing the oracle-web packages, create public synonyms to reference the packages in the user schema in which they were installed. The following example assumes the packages are installed in the *oracleweb* user's schema.

```
create public synonym public_htf for oracleweb.htf;
create public synonym public_htp for oracleweb.htp;
create public synonym public_owa for oracleweb.owa;
create public synonym public_owa_util for oracleweb.owa_util;
create public synonym public_owa_image for oracleweb.owa_image;
```

These packages must also be publicly executable. This can be accomplished by executing the following grant statement:

```
grant execute on htf, htp, owa owa_util, owa_image to public;
```

10.2 Create the DBCGI Accounts and Roles

Create the following oracle accounts with their associated privileges:

- **oramaint** - This is the account which dbcgi uses to modify web usernames and passwords. It should have only the *create session* and *alter user* privileges.

- **oraweb** - This is the generic oracle web user. This account is used to access the database for users who cannot be authenticated via ssl/x509 or who do not have an oracle database account.

  The privileges for this account are application dependent; at a minimum they should include *create session* to allow the oraweb user to connect to the database. Other privileges might include execute permission on application pl/sql procedures.

10.3 Create Database User Web Accounts

Database web user accounts must be created for any user who is permitted individual sign-on via the Web. Web user account names have the form $< username >$_web, where $< username >$ is the user's kerberos principal or regular database login account name. *Note: at this time, the x509 certificates do not contain the client's kerberos principal or account name information. Thus, db.cgi is currently using the client's x500 Common Name (which is part of the x509 certificate) to generate the oracle web account name.*

Privileges granted to web user accounts will vary with the application. At a minimum *create session* and *execute* on certain application pl/sql procedures would be likely privileges.

11 DBCGI Configuration

This section describes other (non-database) configuration issues for the DBCGI program.

11.1 Oracle Environment Variables

Because dBCGI connects to the ORACLE database, it must initialize certain oracle-related environment variables. The DBCGI oracle environment variables are located in a file names *.oraenv* in the same directory as the dbcgi program. The following oracle variables must be initialized in the *.oraenv* file

- **ORACLE_HOME** - Path to the installed oracle binaries and tools.

In addition to oracle-related environment variables, the *.oraenv* also contains the path to the directory where the connect (username/password) information is stored for the *oramaint* and *oraweb* users. This environment variable has the following name:

- **ORACLE_CONFIG_DIR** - full pathname where the *.oramaint* and *.oraweb* files are stored.

The *.orainit* file has the following format: $< parameter >=< value >$. There is only one parameter/value pair per line.

11.2 DBCGI Oracle Login Files

The username/password pairs for the following dbcgi-related oracle users are stored on disk in the $ORACLE_LOGIN_DIR directory. These files have the following names:

- **.oramaintlogin** - username/password (connect string) for the *oramaint* web user maintenance account.
- **.oraweblogin** - username/password (connect string) for the *oraweb* generic oracle web user account.

Each file contains one line with the following format: $<\mathit{username}>/<\mathit{password}>$.