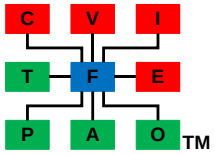


The Data Organization

1251 Yosemite Way
Hayward, CA 94545
(510) 303-8868
rschoenrank@computer.org

Database Requirements
And Requirements Implementation

By Rainer Schoenrank

Data Warehouse Consultant

March 2021

Biography

Rainer Schoenrank is the senior data warehouse consultant for The Data Organization. He has degrees in physics from the University of Victoria and computer science from the University of Victoria and California State University Hayward. He has built data warehouses for clients such as Pacific Bell, Genentech, GE Leasing, SGI, PPFA, Brobeck, BofA, Clorox, Leapfrog and Intuitive Surgical. He can be reached at rschoenrank@computer.org.

Table of Contents

1. DATABASE IMPLEMENTATION REQUIREMENTS

When the logical data model, which contains the business requirements and semantics, is implemented as a database that is used by the business for its record keeping, new functionality must be added to meet data integrity, operational, regulatory and security requirements of the database.

1.1 Data Integrity

The data in the database must be available for use outside of any application silo. This means that the data should adhere to the principles of:

- Data Integrity – For the data to have integrity across all the Data Life Cycle applications, the following must be true:
  - The properties of the data are specified in the data dictionary and the logical data model not in any data sourcing application.
  - The database interface cannot be used to bypass the database security or the data definition.
- Data Independence - For data to be independent of the source application, the following must be true:
  - Changes to the physical location of the database do not require any changes to any applications.
  - Changes to the logical data model do not require any changes to any application.
- Data Accessibility – For data accessibility, the following must be true:
  - All the data is described explicitly in the data dictionary.
  - All the data is accessible to any authorized application.

The most straight forward way to adhere to these principles and to meet the logging, audit and security requirements is to replace the DBMS command line interface for the database with a stored procedure for each application database task on the business objects in the implemented database. The interface choices are described in Database Interface Architecture.

1.2 Application Security

The database security requirements articulate the authentication and authorization objectives to ensure the database can be used by those whose work requires database and data access yet protects the data from others.

When discussing data security, there are several different aspects to the problem, since the database is at the bottom of the processing stack. The security problems of the upper layers of the processing stack are assumed to be solved and the problem of user authentication resides in the Database Management System (DBMS) that encapsulates the database.

DBMS data security assumes that problems of physical, operating system and application security have been addressed and that the DBMS is executing in a secured and trusted environment.

The DBMS does the user authentication. Authentication answers the question usually via a user sign on: Is the user who it says it is? The weakest form of authentication consists of a user identifier and a password.

In a database environment, a user is granted access to individual database objects by the DBMS administrator. The access grant to a user depends on the user type (database owner, application user, application programmer, QA tester or DBA) and the database environment (development, testing, staging or production) that is being used. A model of the authentication policy is shown in the table below:

Database User Role	Database Environment			
	Development	**Testing**	**Staging**	**Production**
Database Administrator	database schema owner	database schema owner	database schema owner	database schema owner
Database Creator	- create/drop table - create stored procedures			
Quality Assurance		- read/write tables - execute stored procedures	- execute stored procedures	
Application Creator	- read/write tables - execute stored procedures			
Application User				- execute stored procedures

Table 1. Authentication Policy

This policy is based on a secure, encapsulated database using a stored procedure interface for data access, error logging, auditing and authorization compliance.

1.3 Database Operations

In order to facilitate finding and fixing errors in the database operations, the database will record all database errors that occur in during the execution of the interface stored procedures.

1.4 Regulatory Compliance (SOX)

In recent years, especially with the passage of the Sarbanes-Oxley Act of 2002, organizations have been forced to tighten up what they report and provide proof that the reported numbers are accurate, complete, and untampered with. The whole tenor of reporting the numbers used as the basis of revenue forecasting and financial status has become much more serious for everyone.

Most IT departments interested in meeting the compliance requirements will try to err on the conservative side. Sarbanes Oxley is not the only set of requirements. Depending on where you do business, there are probably a dozen overlapping financial reporting statutes with similar requirements. A conservative approach to meeting most compliance requirements would suggest the ability to:

- Prove the security of the data copies over time, both online and offline
- Show all end user and administrative accesses of selected data (access auditing)
- Show who created the data and when the data was created (change auditing)
- Show who modified the data and when the data was modified (change auditing)

1.5 Database Auditing

The functionality to record the data for the audit tables would reside in the stored procedure interface used by the applications to access the database.

The reporting and usage auditing requirements are outlined in the Sarbanes-Oxley (SOX) Act.

- Access Audit – the database is required to create a new record each time a database business object is accessed and the record will contain who, when and why the database business object was accessed.
- Change Audit – the database is required to create a new record each time a database business object is changed, recording who and when the database business object is changed.

1.6 Data Authorization Access

The individual database within the DBMS determines which interface stored procedures the user is allowed to execute. This is determined by the row entries in the authorization database tables shown in section 4.6. The Authorization stored procedure answers the question: Is this user allowed to use the requested interface stored procedure?

1.7 Column Data Access

The interface stored procedure determines whether the user accesses a business object (multiple related tables), a table or a view (projection) of the table with in the database. This is the user's column access restriction.

1.8 Row Data Access

The value of the user's Data Ownership Group gives the user access to the subset of the data that belongs to the Group. The value of the user's Data Restriction Level gives the user access to rows with that restriction level or lower. This is the user's row access restriction.

2. DATABASE SERVICE LAYERS

Using the interface stored procedures to access the database (Data as a Service) encapsulates the nonbusiness requirements as separate logical layers within the database. The business data layer (implemented logical data model) interface stored procedure first ensures the security of the data. If the interface request passes the security requirements, then application request is audited and the business data is gathered. If any errors are encountered during the processing, the error logging is invoked, giving the logical structure database structure shown in Figure 1.

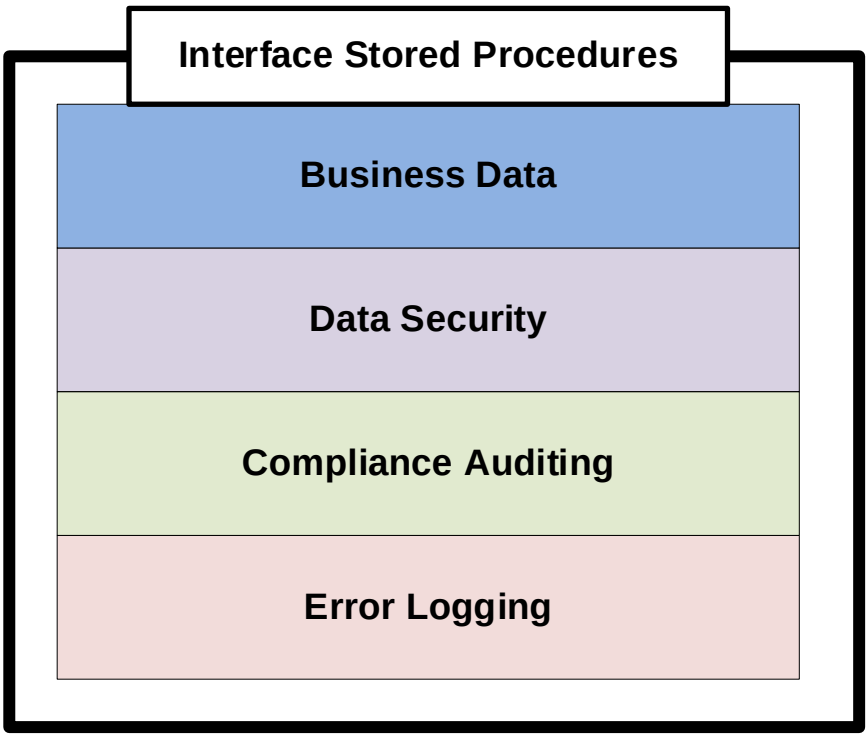


Figure 1. Data as a Service Layers

The order of the layers presented is determined by the processes that the stored procedure interface must execute in order to allow the application user to retrieve and update the correct business data.

Each of the layers encapsulates its data structures, presents a set of stored procedures that allows access to the data in the data structures and does not require the data structures of any of the other layers.

The functionality and data tables required for each layer in the database is shown in Table 2 below.

Database Layer	Database Requirement	Data Tables	Stored Procedures
Business Data	row security column security	ELDM Tables Reference Tables	For each application business object, there is a stored procedure to: - insert a new object - update an existing object - find a single object - delete an existing object - undelete a deleted object - find a set of objects
Data Security	user authorization data security	User Table User Authorization History Table Stored Procedure Table	Read the user's current authorization from table User Authorization History
Compliance Auditing	SOX access logging change logging	Data Access Log Table	insert a new record into the table Access Log
Error Logging	error handling	Error Message Log Table	insert a new record into the table Error Message Log

Table 2 – Data as a Service Layer Functionality

3. IMPLEMENTATION

This chapter details how the database requirements can be implemented. The first section shows a read stored procedure example and how the requirements for error logging, compliance auditing and authorization are met.

The next three sections show how the interface for error logging, compliance auditing and authorization is implemented in the stored procedures and the required tables to support the functionality.

3.1 Interface Stored Procedure Example

The example stored procedure documented below shows how the error logging, audit logging and user authorization verification (highlighted) is done using specialized stored procedures that are an integral part of the requirements layers of the database and are detailed in the next sections.

```
CREATE OR REPLACE FUNCTION DD_BM_BUS_PROC_FN_RD
   --
   -- read a record from the table Business Process
   --
   -- required input parameters:
   --    all key columns
   -- standard input parameters:
   --    name of the application function that called the web service
   --    name of the web service that called the interface procedure
   --    name of the user id that requested the data access
   --
   -- requested output parameters:
   --    all data columns
   --    row meta data modified date
   -- standard output parameters:
   --    error status  - zero if successful, nonzero if not successful
   --    error message - the description of the failure
   --
   -- modified date - 10/4/2020
   -- programmatically generated
   -- © Copyright The Data Organization
   --
(
   I_PROC_KEY IN NUMBER,
   I_PROC_NM IN VARCHAR2,
```

```
   I_APPL_FUNC_NM2 IN VARCHAR2,
   I_WEB_SRVC_NM2 IN VARCHAR2,
   I_USR_NM2 IN VARCHAR2,
   O_PROC_TYP_CD OUT VARCHAR2,
   O_DATA_STWRD_TITL_CD OUT VARCHAR2,
   O_DATA_STWRD_FRST_NM OUT VARCHAR2,
   O_DATA_STWRD_NK_NM OUT VARCHAR2,
   O_DATA_STWRD_MDL_NM OUT VARCHAR2,
   O_DATA_STWRD_LAST_NM OUT VARCHAR2,
   O_DESCR OUT VARCHAR2,
   O_MODF_DT OUT DATE,
   O_STATUS2 OUT NUMERIC,
   O_ERR_MSG2 OUT VARCHAR2
)
   RETURN NUMBER -- return the execution status of the function

   AS

   -- error logging variables (the logging procedure must be available)
   V_RCRD_KEY VARCHAR2(100);
   V_DESCR VARCHAR2(100);
   V_PRCS_NM VARCHAR2(50);
   V_STATUS NUMERIC;
   V_MSG VARCHAR2(100);

   -- exception handling variables
   V_RECORD_COUNT NUMERIC;
   V_NOT_SIGNED EXCEPTION;
   V_NOT_ALLOWED EXCEPTION;
   V_RECORD_CHANGE EXCEPTION;
   V_RECORD_NOT_EXISTS EXCEPTION;
   V_EXIT EXCEPTION;
   V_RETURN_CODE  NUMERIC;

   -- security handling variables
   V_RSTR_LVL NUMERIC;
   V_ROLE_ID VARCHAR2(100);
   V_DATA_OWN_GRP VARCHAR2(4);
   V_APPL_FUNC_NM VARCHAR2(100);

BEGIN
   -- set key value data
   V_RCRD_KEY := I_PROC_KEY
      || ' ' || I_PROC_NM
   ;
   -- set procedure name
   V_PRCS_NM := 'DD_BM_BUS_PROC_FN_RD';
   V_DESCR := 'Step 1 - read';

   -- check that the database access is signed
   IF (I_APPL_FUNC_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;
   IF (I_WEB_SRVC_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;
```



```
   IF (I_USR_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;

   -- check that the user is allowed to use the application function
      V_RETURN_CODE :=  DD_AU_AUTH_CHK_FN_RD(I_APPL_FUNC_NM2,
I_WEB_SRVC_NM2, I_USR_NM2, V_RSTR_LVL, V_ROLE_ID, V_DATA_OWN_GRP,
V_STATUS, V_MSG);

   IF (V_STATUS <> 0) THEN RAISE V_NOT_ALLOWED; END IF;

   -- log database access
   DD_AD_ACS_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, V_DESCR,
I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2, V_STATUS, V_MSG);


   -- find current value of record
   SELECT
      PROC_TYP_CD,
      DATA_STWRD_TITL_CD,
      DATA_STWRD_FRST_NM,
      DATA_STWRD_NK_NM,
      DATA_STWRD_MDL_NM,
      DATA_STWRD_LAST_NM,
      DESCR,
      RMD_MODF_DT
     INTO
      O_PROC_TYP_CD,
      O_DATA_STWRD_TITL_CD,
      O_DATA_STWRD_FRST_NM,
      O_DATA_STWRD_NK_NM,
      O_DATA_STWRD_MDL_NM,
      O_DATA_STWRD_LAST_NM,
      O_DESCR,
      O_MODF_DT
     FROM DD_BM_BUS_PROC
    WHERE (RMD_IN_USE_IND = 'Y')
         AND (RMD_RSTR_LVL <= V_RSTR_LVL)
         AND (RMD_DATA_OWN_GRP  = V_DATA_OWN_GRP)
      AND (PROC_KEY = I_PROC_KEY)
      AND (PROC_NM = I_PROC_NM)
   ;

   -- set output message to success
   O_STATUS2 := 0;
   O_ERR_MSG2 := '';

RETURN 0; -- the function executed normally

EXCEPTION
   WHEN NO_DATA_FOUND
      THEN BEGIN
      O_STATUS2 := -1;
      O_ERR_MSG2 := 'No Data found in the table';
```



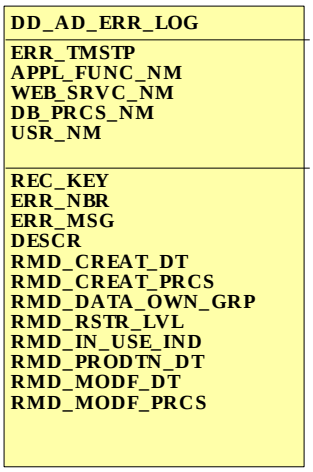
```
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN DUP_VAL_ON_INDEX
      THEN BEGIN
      O_STATUS2 := -2;
      O_ERR_MSG2 := 'The record already exists';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_RECORD_NOT_EXISTS
      THEN BEGIN
      O_STATUS2 := -3;
      O_ERR_MSG2 := 'Record does not exist';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_RECORD_CHANGE
      THEN BEGIN
      O_STATUS2 := -4;
      O_ERR_MSG2 := 'Record data has changed since last read';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN TOO_MANY_ROWS
      THEN BEGIN
      O_STATUS2 := -5;
      O_ERR_MSG2 := 'More than one record was found';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_NOT_SIGNED
```



```
      THEN BEGIN
      O_STATUS2 := -6;
      O_ERR_MSG2 := 'Database access is not signed by calling
procedure';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_NOT_ALLOWED
      THEN BEGIN
      O_STATUS2 := -7;
      O_ERR_MSG2 := 'Database access is not authorized for this user';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_EXIT
      THEN BEGIN
      O_STATUS2 :=  0;
      O_ERR_MSG2 := '';
      RETURN 0; -- the function executed normally
      END;

   WHEN OTHERS
      THEN BEGIN
      O_STATUS2 := SQLCODE;
      O_ERR_MSG2 := SQLERRM;
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_WEB_SRVC_NM2, I_USR_NM2,
V_STATUS, V_MSG);
      -- signal error to calling program
      RAISE_APPLICATION_ERROR(-20000, SQLERRM);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

END DD_BM_BUS_PROC_FN_RD;
/
```


3.2 Error Logging Table

The table used for recording the errors encountered the database is shown below. It contains the error time stamp, the application function requesting the access, the web service providing the database access, the interface stored procedure name, the user name, the key of the record being accessed, the error number, and the error message.



3.3 Error Logging Stored Procedure

The database stored procedure that stores all the unhandled exceptions (errors) of an interface stored procedures is listed below:

```
CREATE OR REPLACE PROCEDURE DD_AD_ERR_LOG_SP_CRT
   --
   -- insert a new record into the table Error Message Log
   --
   -- required input parameters:
   --    name of database interface procedure doing the error log
   --    key of the record
   --    SQL error number
   --    SQL error message
   --    description provided by calling procedure
   -- standard input parameters:
   --    name of application function that called the web service
   --    name of web service that called the interface procedure
   --    name of the user id that requested the data access
   --
   -- requested output parameters:
   --    none
   -- standard output parameters:
   --    error status  - zero if successful, nonzero if not successful
   --    error message - the description of the failure
   --
   -- modified date - 10/25/2011
   -- © Copyright The Data Organization
   --
(
   I_CALL_PRCS IN VARCHAR2,
   I_REC_KEY IN VARCHAR2,
   I_ERR_NBR IN NUMBER,
   I_ERR_MSG IN VARCHAR2,
   I_DESCR IN VARCHAR2,
   I_APPL_FUNC_NM2 IN VARCHAR2,
   I_WEB_SRVC_NM2 IN VARCHAR2,
   I_USR_NM2 IN VARCHAR2,
   O_STATUS2 OUT NUMBER,
   O_ERR_MSG2 OUT VARCHAR2
)
   AS

   -- error logging variables (the logging procedure must be available)
   V_RCRD_KEY VARCHAR2(100);
   V_DESCR VARCHAR2(100);
   V_PRCS_NM VARCHAR2(50);
```

```
   V_STATUS NUMERIC;
   V_MSG VARCHAR2(100);

   -- exception handling variables
   V_RECORD_COUNT NUMERIC;
   V_NOT_SIGNED EXCEPTION;
   V_NOT_ALLOWED EXCEPTION;
   V_RECORD_CHANGE EXCEPTION;
   V_RECORD_NOT_EXISTS EXCEPTION;
   V_EXIT EXCEPTION;
   V_RETURN_CODE NUMERIC;

BEGIN
   -- set procedure name
   V_PRCS_NM := 'DD_AD_ERR_LOG_SP_CRT';
   V_DESCR := 'Step 1 - insert';

   -- check that the database access is signed
   IF (I_APPL_FUNC_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;
   IF (I_WEB_SRVC_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;
   IF (I_USR_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;

   -- check whether the requested new record exists is not required

   -- insert new record
   INSERT INTO DD_AD_ERR_LOG (
      ERR_TMSTP,
      APPL_FUNC_NM,
      WEB_SRVC_NM,
      DB_PRCS_NM,
      USR_NM,
      REC_KEY,
      ERR_NBR,
      ERR_MSG,
      DESCR,
      RMD_CREAT_DT,
      RMD_CREAT_PRCS,
      RMD_DATA_OWN_GRP,
      RMD_RSTR_LVL,
      RMD_IN_USE_IND,
      RMD_PRODTN_DT,
      RMD_MODF_DT,
      RMD_MODF_PRCS
   ) VALUES (
      SYSTIMESTAMP,
      I_APPL_FUNC_NM2,
      I_WEB_SRVC_NM2,
      I_CALL_PRCS,
      I_USR_NM2,
      I_REC_KEY,
      I_ERR_NBR,
      I_ERR_MSG,
```



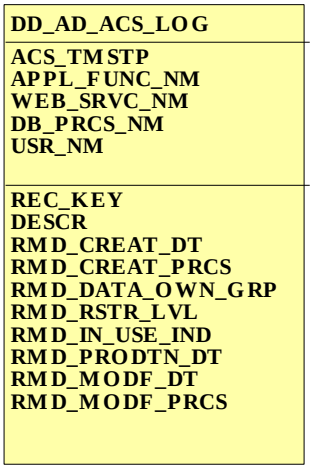
```
      I_DESCR,
      SYSDATE,
      I_CALL_PRCS,
      0,
      0,
      'Y',
      NULL,
      SYSDATE,
      I_CALL_PRCS
   );

    -- set output message to success
    O_STATUS2 := 0;
    O_ERR_MSG2 := '';

EXCEPTION
   WHEN NO_DATA_FOUND
      THEN BEGIN
      O_STATUS2 := -1;
      O_ERR_MSG2 := 'No records found in the table';
      END;

   WHEN DUP_VAL_ON_INDEX
      THEN BEGIN
      O_STATUS2 := -2;
      O_ERR_MSG2 := 'The record already exists';
      END;

   WHEN V_RECORD_NOT_EXISTS
      THEN BEGIN
      O_STATUS2 := -3;
      O_ERR_MSG2 := 'Record does not exist';
      END;

   WHEN V_RECORD_CHANGE
      THEN BEGIN
      O_STATUS2 := -4;
      O_ERR_MSG2 := 'Record data has changed since last read';
      END;

   WHEN TOO_MANY_ROWS
      THEN BEGIN
      O_STATUS2 := -5;
      O_ERR_MSG2 := 'More than one record was found';
      END;

   WHEN V_NOT_SIGNED
      THEN BEGIN
      O_STATUS2 := -6;
      O_ERR_MSG2 := 'Database access is not signed by calling
procedure';
      END;
```



```
   WHEN V_NOT_ALLOWED
      THEN BEGIN
      O_STATUS2 := -7;
      O_ERR_MSG2 := 'Database access is not authorized for this user';
      END;

   WHEN V_EXIT
      THEN BEGIN
      O_STATUS2 := 0;
      O_ERR_MSG2 := '';
      END;

   WHEN OTHERS
      THEN BEGIN
      O_STATUS2 := SQLCODE;
      O_ERR_MSG2 := SQLERRM;
      END;

END DD_AD_ERR_LOG_SP_CRT;
/
```


3.4 Compliance Auditing Table

The table used for recording the accesses to the data in the database is shown below. It contains the access time stamp, the application function requesting the access, the web service providing the database access, the interface stored procedure name, the user name and the key of the record being accessed.



3.5 Compliance Auditing Stored Procedure

The database stored procedure that writes a new record into the audit table is listed below:

```
CREATE OR REPLACE PROCEDURE DD_AD_ACS_LOG_SP_CRT
   --
   -- insert a new record into the table Access Log
   --
   -- required input parameters:
   --    name of database interface procedure doing the data access
   --    key of the record
   --    description provided by calling procedure
   -- standard input parameters:
   --    name of application function that called the web service
   --    name of web service that called the interface procedure
   --    name of the user id that requested the data access
   --
   -- requested output parameters:
   --    none
   -- standard output parameters:
   --    error status  - zero if successful, nonzero if not successful
   --    error message - the description of the failure
   --
   -- modified date - 10/25/2011
   -- © Copyright The Data Organization
   --
(
   I_CALL_PRCS IN VARCHAR2,
   I_REC_KEY IN VARCHAR2,
   I_DESCR IN VARCHAR2,
   I_APPL_FUNC_NM2 IN VARCHAR2,
   I_WEB_SRVC_NM2 IN VARCHAR2,
   I_USR_NM2 IN VARCHAR2,
   O_STATUS2 OUT NUMBER,
   O_ERR_MSG2 OUT VARCHAR2
)

   AS

   -- error logging variables (the logging procedure must be available)
   V_RCRD_KEY VARCHAR2(100);
   V_DESCR VARCHAR2(100);
   V_PRCS_NM VARCHAR2(50);
   V_STATUS NUMERIC;
   V_MSG VARCHAR2(100);

   -- exception handling variables
```

```
   V_RECORD_COUNT NUMERIC;
   V_NOT_SIGNED EXCEPTION;
   V_NOT_ALLOWED EXCEPTION;
   V_RECORD_CHANGE EXCEPTION;
   V_RECORD_NOT_EXISTS EXCEPTION;
   V_EXIT EXCEPTION;
   V_RETURN_CODE NUMERIC;

BEGIN
   -- set procedure name
   V_PRCS_NM := 'DD_AD_ACS_LOG_SP_CRT';
   V_DESCR := 'Step 1 - insert';

   -- check that the database access is signed
   IF (I_APPL_FUNC_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;
   IF (I_WEB_SRVC_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;
   IF (I_USR_NM2 IS NULL) THEN RAISE V_NOT_SIGNED; END IF;

   -- check whether the requested new record exists is not required

   -- insert new record
   INSERT INTO DD_AD_ACS_LOG (
      ACS_TMSTP,
      APPL_FUNC_NM,
      WEB_SRVC_NM,
      DB_PRCS_NM,
      USR_NM,
      REC_KEY,
      DESCR,
      RMD_CREAT_DT,
      RMD_CREAT_PRCS,
      RMD_DATA_OWN_GRP,
      RMD_RSTR_LVL,
      RMD_IN_USE_IND,
      RMD_PRODTN_DT,
      RMD_MODF_DT,
      RMD_MODF_PRCS
   ) VALUES (
      SYSTIMESTAMP,
      I_APPL_FUNC_NM2,
      I_WEB_SRVC_NM2,
      I_CALL_PRCS,
      I_USR_NM2,
      I_REC_KEY,
      I_DESCR,
      SYSDATE,
      I_CALL_PRCS,
      0,
      0,
      'Y',
      NULL,
      SYSDATE,
```



```
      I_CALL_PRCS
   );

    -- set output message to success
    O_STATUS2 := 0;
    O_ERR_MSG2 := '';

EXCEPTION
   WHEN NO_DATA_FOUND
      THEN BEGIN
      O_STATUS2 := -1;
      O_ERR_MSG2 := 'No records found in the table';
      END;

   WHEN DUP_VAL_ON_INDEX
      THEN BEGIN
      O_STATUS2 := -2;
      O_ERR_MSG2 := 'The record already exists';
      END;

   WHEN V_RECORD_NOT_EXISTS
      THEN BEGIN
      O_STATUS2 := -3;
      O_ERR_MSG2 := 'Record does not exist';
      END;

   WHEN V_RECORD_CHANGE
      THEN BEGIN
      O_STATUS2 := -4;
      O_ERR_MSG2 := 'Record data has changed since last read';
      END;

   WHEN TOO_MANY_ROWS
      THEN BEGIN
      O_STATUS2 := -5;
      O_ERR_MSG2 := 'More than one record was found';
      END;

   WHEN V_NOT_SIGNED
      THEN BEGIN
      O_STATUS2 := -6;
      O_ERR_MSG2 := 'Database access is not signed by calling
procedure';
      END;

   WHEN V_NOT_ALLOWED
      THEN BEGIN
      O_STATUS2 := -7;
      O_ERR_MSG2 := 'Database access is not authorized for this user';
      END;

   WHEN V_EXIT
```

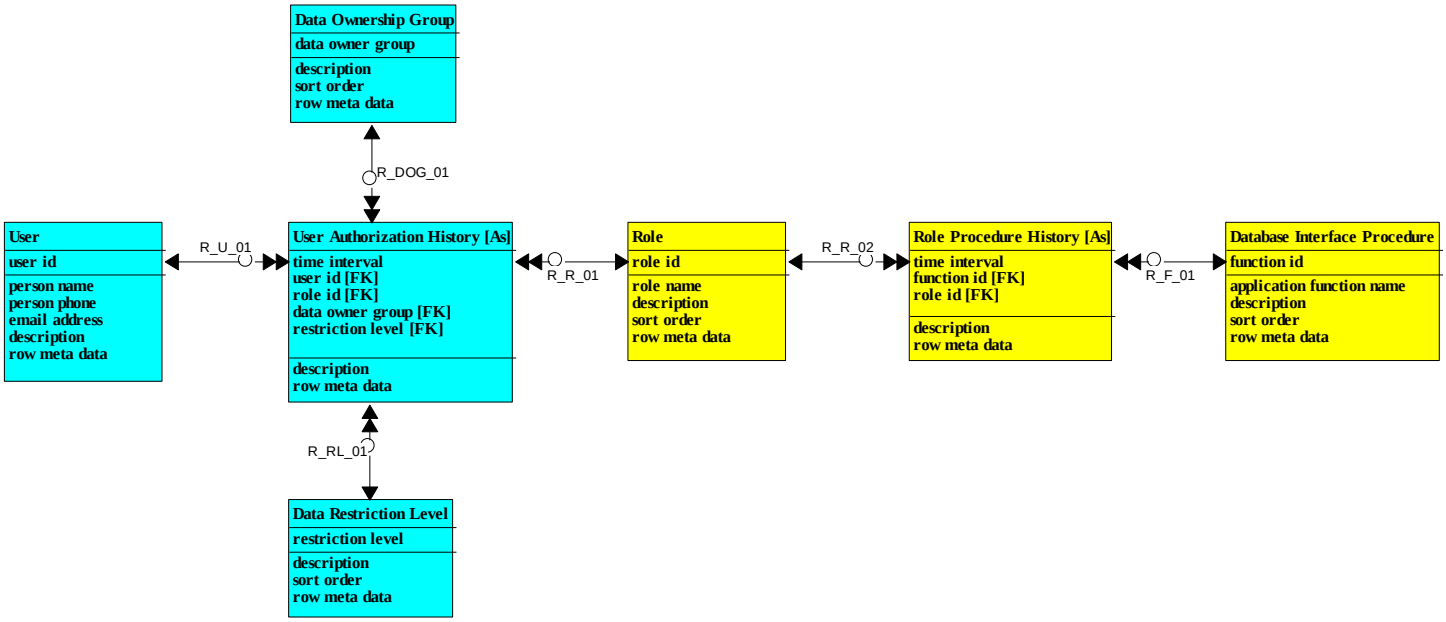


```
      THEN BEGIN
      O_STATUS2 := 0;
      O_ERR_MSG2 := '';
      END;

   WHEN OTHERS
      THEN BEGIN
      O_STATUS2 := SQLCODE;
      O_ERR_MSG2 := SQLERRM;
      END;

END DD_AD_ACS_LOG_SP_CRT;
/
```


3.6 User Authorization Tables

The tables used for the user authorization is shown in the diagram below. The tables include an inventory of the database interface procedures which encapsulate the application business objects and a database role table that lists all the interfaces procedures that are used by a business role, such as accounts receivable manager, etc.

The user is assigned one or more roles for various periods of time (User Authorization History) as his position requires.



3.7 User Authorization Stored Procedure

The database stored procedure that validates (or not) that the user is authorized to execute the requested functionality on the database is listed below:

```
CREATE OR REPLACE FUNCTION DD_AU_AUTH_CHK_FN_RD
   --
   -- Read the user's current authorization from table User
Authorization History
   --
   -- required input parameters:
   --    none
   -- standard input parameters:
   --    name of the application function that called the web service
   --    name of the web service that called the database interface
procedure
   --    name of the user id that requested the data access
   --
   -- requested output parameters:
   --    user's role
   --    user's data ownership group
   --    user's restriction level
   -- standard output parameters:
   --    error status  - zero if successful, nonzero if not successful
   --    error message - the description of the failure
   --
   -- modified date - 10/27/2011
   -- © Copyright The Data Organization
   --
(
   I_APPL_FUNC_NM2 IN VARCHAR2,
   I_PRCS_NM2 IN VARCHAR2,
   I_USR_NM2 IN VARCHAR2,
   O_ROLE_ID OUT VARCHAR2,
   O_DATA_OWN_GRP OUT VARCHAR2,
   O_RSTR_LVL OUT NUMERIC,
   O_STATUS2 OUT NUMERIC,
   O_ERR_MSG2 OUT VARCHAR2
)
   RETURN NUMBER -- return the execution status of the function

   AS

   -- error logging variables (the logging procedure must be available)
   V_RCRD_KEY VARCHAR2(100);
   V_DESCR VARCHAR2(100);
   V_PRCS_NM VARCHAR2(50);
```

```
   V_STATUS NUMERIC;
   V_MSG VARCHAR2(100);

   -- exception handling variables
   V_RECORD_COUNT NUMERIC;
   V_NOT_SIGNED EXCEPTION;
   V_NOT_ALLOWED EXCEPTION;
   V_RECORD_CHANGE EXCEPTION;
   V_RECORD_NOT_EXISTS EXCEPTION;
   V_EXIT EXCEPTION;
   V_RETURN_CODE  NUMERIC;

BEGIN
   -- set procedure name
   V_PRCS_NM := 'DD_AU_AUTH_CHKC_FN_RD';

   -- set key value data
   V_RCRD_KEY := I_USR_NM2
   ;

   -- get the user's role, data group and restriction level
   V_DESCR := 'Step 1 - get user credentials';
   SELECT
      ROLE_ID,
      DATA_OWN_GRP,
      RSTR_LVL
     INTO
      O_ROLE_ID,
      O_DATA_OWN_GRP,
      O_RSTR_LVL
     FROM DD_AU_USR
          INNER JOIN DD_AU_USR_AUTH_HIST
             ON DD_AU_USR.USR_ID = DD_AU_USR_AUTH_HIST.USR_ID
    WHERE (DD_AU_USR.RMD_IN_USE_IND = 'Y')
      AND (UPPER(DD_AU_USR.USR_ID) = UPPER(I_USR_NM2))
      AND (TIME_INTVL_STRT_DT  <= SYSDATE)
      AND (TIME_INTVL_END_DT   >= SYSDATE)
   ;

   -- find current authorization
   V_DESCR := 'Step 2 - validate user authorization';
   SELECT
      COUNT(ROLE_ID)
     INTO
      V_RECORD_COUNT
     FROM DD_AU_APPL_FUNC
          INNER JOIN DD_AU_ROLE_FUNC_HIST
             ON DD_AU_ROLE_FUNC_HIST.FUNC_ID = DD_AU_APPL_FUNC.FUNC_ID
    WHERE ((DD_AU_APPL_FUNC.RMD_IN_USE_IND) = 'Y')
      AND (DD_AU_APPL_FUNC.APPL_FUNC_NM = I_APPL_FUNC_NM2)
      AND (DD_AU_ROLE_FUNC_HIST.ROLE_ID = O_ROLE_ID)
      AND (DD_AU_ROLE_FUNC_HIST.TIME_INTVL_STRT_DT <= SYSDATE)
```



```
      AND (DD_AU_ROLE_FUNC_HIST.TIME_INTVL_END_DT  >= SYSDATE)
   ;

   IF (V_RECORD_COUNT = 0) THEN RAISE V_NOT_ALLOWED; END IF;
   IF (V_RECORD_COUNT = 1) THEN RAISE V_EXIT; END IF;
   IF (V_RECORD_COUNT > 1) THEN RAISE TOO_MANY_ROWS; END IF;

RETURN 0;   -- the function executed normally

EXCEPTION
   WHEN NO_DATA_FOUND
      THEN BEGIN
      O_STATUS2 := -1;
      O_ERR_MSG2 := 'No Data found in the table';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN DUP_VAL_ON_INDEX
      THEN BEGIN
      O_STATUS2 := -2;
      O_ERR_MSG2 := 'The record already exists';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_RECORD_NOT_EXISTS
      THEN BEGIN
      O_STATUS2 := -3;
      O_ERR_MSG2 := 'Record to delete does not exist';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_RECORD_CHANGE
      THEN BEGIN
      O_STATUS2 := -4;
      O_ERR_MSG2 := 'Record data has changed since last read';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
```



```
      END;

   WHEN TOO_MANY_ROWS
      THEN BEGIN
      O_STATUS2 := -5;
      O_ERR_MSG2 := 'More than one record was found';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_NOT_SIGNED
      THEN BEGIN
      O_STATUS2 := -6;
      O_ERR_MSG2 := 'Database access is not signed by calling
procedure';
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_NOT_ALLOWED
      THEN BEGIN
      O_STATUS2 := -7;
      O_ERR_MSG2 := 'Database access is not authorized for this user';
      RETURN -1;  -- an error occurred in the execution of the function
      END;

   WHEN V_EXIT
      THEN BEGIN
      O_STATUS2 := 0;
      O_ERR_MSG2 := '';
      RETURN 0;
      END;

   WHEN OTHERS
      THEN BEGIN
      O_STATUS2 := SQLCODE;
      O_ERR_MSG2 := SQLERRM;
      -- log error message
      DD_AD_ERR_LOG_SP_CRT(V_PRCS_NM, V_RCRD_KEY, O_STATUS2,
O_ERR_MSG2, V_DESCR, I_APPL_FUNC_NM2, I_PRCS_NM2, I_USR_NM2, V_STATUS,
V_MSG);
      -- signal error to calling program
      RAISE_APPLICATION_ERROR(-20000, SQLERRM);
      END;

END DD_AU_AUTH_CHK_FN_RD;
```


/

