INTRODUCTION

DBMS_METADATA was introduced with Oracle9i with the promise to easily extract the DDL for an object, schema, or entire database with simple calls. No longer did you have to perform an export or import to get poorly formatted DDL (and then have to reformat it to actually get it to work). A supplied package would do it all for you . . . if it worked properly. Unfortunately, DBMS_METADATA developed a well-deserved reputation for being poorly documented, difficult to work with, and having more than its fair share of bugs. This session will examine the basic calls, workarounds for several bugs, and how to put it all together as a flexible script.

Table of Contents

WHAT IS DBMS_METADATA?

The DBMS_METADATA package contains 13 subprograms that can be used to extract ddl from a database. It can be used to extract the ddl in either textual or XML format. I have only worked with the textual format (I'm a bitter old command line dba). It can be used to extract a single object's metadata or the ddl for a whole schema (with a little work and creativity).

It was originally introduced in 9i and more options were added in 10g. I have not used 11g yet (not an early adopter, sorry). I first worked with it on 9iR2 on Solaris. One main problem I found was that when I moved to the same release on a different platform or a different patch set on the same platform the behavior changed.

Why use it?

The original reason to use dbms_metadata was to extract the ddl for a project team (see the case study later). The papers I found on line mentioned how useful and easy it was to work with, complete with examples. As part of the standard installation, no special scripts were required. Because the calls were supposed to be rather simple, all the project team needed would be a sql*plus connection and the ability to spool output. We also knew that this was not a one time request, so we wanted a repeatable process.

Documentation and Papers

One of the challenges for me (as a lowly DBA) was the documentation. I think it was written by a very smart programmer who learns by reading specifications. Not so good for me because I learn by example. I also found quite a few errors and incomplete information in the documentation. Read the documentation with a skeptical eye, always test what is written.

Expected Behavior and Bugs

At last check, there were over 200 entries in the bug database for dbms_metadata. Fortunately, none of them appear to crash the database or corrupt data. However, there are lots of platform/release specific bugs, so a workaround for 9iR2 on Solaris may not work for 9iR2 on Windows.

Parameter Requirements

There are two things to keep in mind when using dbms_metadata

Parameter are case sensitive

```
Correct
dbms_metadata.get_ddl('VIEW','C_VIEW','OE')
Error
dbms_metadata.get_ddl('view','C_VIEW','OE')
 ERROR:
ORA-31600: invalid input value view for parameter OBJECT_TYPE in function
                GET_DDL
ORA-06512: at "SYS.DBMS_METADATA", line 2681
ORA-06512: at "SYS.DBMS_METADATA", line 2732
ORA-06512: at "SYS.DBMS_METADATA", line 4333
ORA-06512: at line 1
```

Parameter passing is by position only

```
Correct
dbms_metadata.get_ddl('VIEW','C_VIEW','OE')
Error
dbms_metadata.get_ddl(object_type=>'VIEW',name=>'C_VIEW',schema=>'OE')
                                 *
ERROR at line 2:
ORA-00907: missing right parenthesis
```

Security and Permissions

EXECUTE on DBMS_METADATA is granted to PUBLIC, so any user that can connect to the database can run the package. There is also a public synonym, so you do not need to qualify the call with the owner. I have not used or tested the package inside a PL/SQL package. If you choose to do so, expect some problems.

```
select * from dba_tab_privs
where table_name = 'DBMS_METADATA'
GRANTEE OWNER TABLE_NAME    GRANTOR PRIVILEGE GRA HIE
------- ----- ------------- ------- --------- --- ---
PUBLIC SYS    DBMS_METADATA SYS     EXECUTE   NO  NO
```

Nonprivileged User

A user can always retrieve the metadata for their own objects. They can also retrieve metadata for PUBLIC synonyms and privileges, system privileges granted, and object privileges granted to them or by them to others. For objects where the user is granted permissions, they can retrieve the grant command, but cannot retrieve other metadata related to the object.

According to the documentation, if the user attempts to retrieve metadata for an object that they do not have privilege on, the metadata is not output and no error is returned. However, in testing, I find that this is not the case.

connect hr/demo

```
GRANT select, insert, update, delete, alter ON locations TO oe;
connect oe/demo;

SELECT     DBMS_METADATA.GET_DDL('TABLE', 'LOCATIONS', 'HR') ddl_call
FROM       dual;
ERROR:
ORA-31603: object "LOCATIONS" of type TABLE not found in schema "HR"
ORA-06512: at "SYS.DBMS_SYS_ERROR", line 105
ORA-06512: at "SYS.DBMS_METADATA", line 2805
ORA-06512: at "SYS.DBMS_METADATA", line 4333
ORA-06512: at line 1
```

Privileged User

A privileged user can retrieve the metadata for objects in another schema. If the user has SELECT_CATALOG_ROLE, they are considered a privileged user. Most dba level accounts have this role.

SET UP

In order to output the metadata in usable format, there is some minor set up items to address.

*SQL*Plus settings*

LINESIZE	**Wide enough to make the output readable. I usually set it to 125.**
PAGESIZE	**0 to suppress all heading information**
FEEDBACK	**OFF to suppress rows information**
LONG	**At least 100000 for the CLOB output**
TRIMSPOOL	**ON to trim any following spaces**
COLUMN	**WORD_WRAP to insure breaks are not in the middle of words. This also suppresses some of the formatting used by DBMS_METADATA.**

DBMS_METADATA settings

SET_TRANSFORM_PARAM

This function controls metadata output format. There are a series of parameters that can be set to TRUE or FALSE.

```
dbms_metadata.set_transform_param(transform_handle,'PARAMETER_NAME',value)
```

TRANSFORM HANDLE

The documentation states that it is "The handle returned from `OPEN` when this transform is used to retrieve objects. ", which is "The return value is an opaque context handle for the set of objects to be used in subsequent calls". I spent hours trying to determine what this thing was...and then I found out that I really did not need one. I found a reference to (DBMS_METADATA.SESSION_TRANSFORM), which the documentation says is 'an enumerated constant'.

PARAMETER NAME

The name of the parameter you want to change. The documentation is actually very clear. Some of the parameters I use frequently will be presented shortly.

VALUE

Almost all the parameters are boolean, though there is one (PCTSPACE) that is a number.

Default output

The current default output is nearly complete. All this output lacks to recreate the OE.CUSTOMERS table is a sql terminator.

```
  CREATE TABLE "OE"."ORDER_ITEMS"
   (    "ORDER_ID" NUMBER(12,0),
        "LINE_ITEM_ID" NUMBER(3,0) NOT NULL ENABLE,
        "PRODUCT_ID" NUMBER(6,0) NOT NULL ENABLE,
        "UNIT_PRICE" NUMBER(8,2),
        "QUANTITY" NUMBER(8,0),
         CONSTRAINT "ORDER_ITEMS_PK" PRIMARY KEY ("ORDER_ID", "LINE_ITEM_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "EXAMPLE"  ENABLE,
         CONSTRAINT "ORDER_ITEMS_ORDER_ID_FK" FOREIGN KEY ("ORDER_ID")
          REFERENCES "OE"."ORDERS" ("ORDER_ID") ON DELETE CASCADE ENABLE
                 NOVALIDATE,
         CONSTRAINT "ORDER_ITEMS_PRODUCT_ID_FK" FOREIGN KEY ("PRODUCT_ID")
          REFERENCES "OE"."PRODUCT_INFORMATION" ("PRODUCT_ID") ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT)
  TABLESPACE "EXAMPLE"
```

PRETTY

This option formats the ddl with line feeds and indentation. Without it, the ddl usually won't run as breaks will occur in the middle of object names or options (unless you set the COLUMN options properly). It defaults to TRUE, so you won't have to use it. However, it is always a good idea in case the default value changes in a future release.

```
  CREATE TABLE "OE"."ORDER_ITEMS" ("ORDER_ID" NUMBER(12,0), "LINE_ITEM_ID" NUMBE
R(3,0) NOT NULL ENABLE, "PRODUCT_ID" NUMBER(6,0) NOT NULL ENABLE,
                  "UNIT_PRICE" N
UMBER(8,2), "QUANTITY" NUMBER(8,0),  CONSTRAINT "ORDER_ITEMS_PK" PRIMARY KEY ("O
RDER_ID", "LINE_ITEM_ID") USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE
 STATISTICS  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 21474836
45 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAULT) TABLESPACE "
EXAMPLE"  ENABLE,  CONSTRAINT "ORDER_ITEMS_ORDER_ID_FK" FOREIGN KEY ("ORDER_ID")
 REFERENCES "OE"."ORDERS" ("ORDER_ID") ON DELETE CASCADE ENABLE NOVALIDATE,  CON
STRAINT "ORDER_ITEMS_PRODUCT_ID_FK" FOREIGN KEY ("PRODUCT_ID") REFERENCES "OE"."
PRODUCT_INFORMATION" ("PRODUCT_ID") ENABLE) PCTFREE 10 PCTUSED 40 INITRANS 1 MAX
TRANS 255 NOCOMPRESS LOGGING STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAX
EXTENTS 2147483645 PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1 BUFFER_POOL DEFAU
LT) TABLESPACE "EXAMPLE"
```

STORAGE {TRUE | FALSE }

This option controls the STORAGE clause for objects. Most of the systems I have worked with for the past 9 years use the tablespace default settings, so I prefer to not have the STORAGE clause output. Note that this does not remove all STORAGE related clauses, as the TABLESPACE clause is still output.

SEGMENT_ATTRIBUTES {TRUE | FALSE }

This suppresses the segment attributes, such as storage, segment information (PCTFREE, PCTUSED, etc.). Unfortunately, it also suppresses the USING INDEX clause as well.

CONSTRAINTS {TRUE | FALSE }

This controls the output of CONSTRAINTS...not all of them. It still outputs the Foreign Key constraint. This can be controlled by the next parameter.

REF_CONSTRAINTS {TRUE | FALSE }

Referential Integrity constraints can be omitted by using this parameter. However, it only deals with referential integrity constraints, all other constraints are left in place.

When retrieving multiple objects, it is a good idea to set this to false to avoid problems with circular referential integrity constraints.

CONSTRAINTS_AS_ALTER {TRUE | FALSE }

This moves the constraint clauses to after the CREATE TABLE statement. If you have not specified SQLTERMINATOR, the resulting output will be a series of statements without any terminator. Not very pretty at all!

SQLTERMINATOR {TRUE | FALSE }

This setting enables you to append a sqlterminator (;) to the end of the call. The documentation states that the value can be either a ; or /, but in practice, the only character that was ever output is the ;. If you want to use the /, there are ways as we will see later.

One of the bugs encountered was that the sqlterminator was not output for Referential Integrity constraints.

FORCE {TRUE | FALSE }

This includes the FORCE option for CREATE VIEW syntax.

DEFAULT {TRUE }

This will return the transform parameters back to the DEFAULT values.

My preferred output

If I am recreating a single object or a small number, I prefer this format. If there are a large number of objects, the proper order is not guaranteed, so I like to have referential integrity constraints separate.

```
dbms_metadata.set_transform_param(dbms_metadata.session_transform,'SQLTERMINATOR',true);
dbms_metadata.set_transform_param(dbms_metadata.session_transform,'STORAGE',false);
DDL_CALL
--------------------------------------------------------------------------------
  CREATE TABLE "OE"."ORDER_ITEMS"
   (    "ORDER_ID" NUMBER(12,0),
        "LINE_ITEM_ID" NUMBER(3,0) NOT NULL ENABLE,
        "PRODUCT_ID" NUMBER(6,0) NOT NULL ENABLE,
        "UNIT_PRICE" NUMBER(8,2),
        "QUANTITY" NUMBER(8,0),
         CONSTRAINT "ORDER_ITEMS_PK" PRIMARY KEY ("ORDER_ID", "LINE_ITEM_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  TABLESPACE "EXAMPLE"  ENABLE,
         CONSTRAINT "ORDER_ITEMS_ORDER_ID_FK" FOREIGN KEY ("ORDER_ID")
          REFERENCES "OE"."ORDERS" ("ORDER_ID") ON DELETE CASCADE ENABLE NOVALIDATE,
         CONSTRAINT "ORDER_ITEMS_PRODUCT_ID_FK" FOREIGN KEY ("PRODUCT_ID")
          REFERENCES "OE"."PRODUCT_INFORMATION" ("PRODUCT_ID") ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  TABLESPACE "EXAMPLE" ;
```

RETRIEVING METADATA

Once we have the SQL*Plus and dbms_metadata parameter set up, it is time to actually extract the ddl. To return the ddl in a textual format, the GET_DDL calls are used; for ddl in XML format, the GET_XML calls are used...but not by me.

The main challenge you face is running into the bugs and 'expected behaviors'.

Subprograms

PARAMETERS

There are up to 6 parameters, but in practice I only used 3. And they are pretty self explanatory. The schema defaults to the current user schema.

- OBJECT_TYPE
- OBJECT_NAME
- SCHEMA

RETURN

A CLOB is returned by the procedures, which is why you need to SET LONG when using SQL*Plus. The actual format of the output is influenced by the settings you have set previously.

GET_DDL

This returns the ddl for the object specified.

GET_GRANTED_DDL

This procedure is used to get the ddl for grants for an object or schema, depending on type of privilege.

GET_DEPENDENT_DDL

This returns the dependent ddl for an object.

HOW TO

Setup

```
SET LINESIZE 132 PAGESIZE 0 FEEDBACK off VERIFY off TRIMSPOOL on LONG 1000000
COLUMN ddl_string FORMAT A100 WORD_WRAP
EXEC DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM,'STORAGE',false);
EXEC DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM,'PRETTY',true);
EXEC DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM,'SQLTERMINATOR',true);
EXEC DBMS_METADATA.SET_TRANSFORM_PARAM(DBMS_METADATA.SESSION_TRANSFORM,'REF_CONSTRAINTS',false);
```

Retrieve a Single Object

I am not disabling referential integrity constraints for this example as the table call is self contained.

Table

```
SELECT      dbms_metadata.get_ddl('TABLE','CUSTOMERS') ddl_string
FROM        dual

CREATE TABLE "OE"."CUSTOMERS"
 (    "CUSTOMER_ID" NUMBER(6,0),
      "CUST_FIRST_NAME" VARCHAR2(20) CONSTRAINT "CUST_FNAME_NN" NOT NULL ENABLE,
        "CUST_LAST_NAME" VARCHAR2(20) CONSTRAINT "CUST_LNAME_NN" NOT NULL ENABLE,
        "CUST_ADDRESS" "OE"."CUST_ADDRESS_TYP" ,
        "PHONE_NUMBERS" "OE"."PHONE_LIST_TYP" ,
        "NLS_LANGUAGE" VARCHAR2(3),
        "NLS_TERRITORY" VARCHAR2(30),
        "CREDIT_LIMIT" NUMBER(9,2),
        "CUST_EMAIL" VARCHAR2(30),
        "ACCOUNT_MGR_ID" NUMBER(6,0),
        "CUST_GEO_LOCATION" "MDSYS"."SDO_GEOMETRY" ,
        "DATE_OF_BIRTH" DATE,
        "MARITAL_STATUS" VARCHAR2(20),
        "GENDER" VARCHAR2(1),
        "INCOME_LEVEL" VARCHAR2(20),
         CONSTRAINT "CUSTOMER_CREDIT_LIMIT_MAX" CHECK (credit_limit <= 5000) ENABLE,
         CONSTRAINT "CUSTOMER_ID_MIN" CHECK (customer_id > 0) ENABLE,
         CONSTRAINT "CUSTOMERS_PK" PRIMARY KEY ("CUSTOMER_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  TABLESPACE "EXAMPLE"  ENABLE,
         CONSTRAINT "CUSTOMERS_ACCOUNT_MANAGER_FK" FOREIGN KEY ("ACCOUNT_MGR_ID")
          REFERENCES "HR"."EMPLOYEES" ("EMPLOYEE_ID") ON DELETE SET NULL ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  TABLESPACE "EXAMPLE" ;
```

View

```
SELECT     dbms_metadata.get_ddl('VIEW','OC_CUSTOMERS') ddl_string
FROM       dual
CREATE OR REPLACE FORCE VIEW "OE"."OC_CUSTOMERS" OF "OE"."CUSTOMER_TYP"
WITH OBJECT IDENTIFIER (customer_id) AS
SELECT c.customer_id, c.cust_first_name, c.cust_last_name, c.cust_address,
         c.phone_numbers,c.nls_language,c.nls_territory,c.credit_limit,
         c.cust_email,
         CAST(MULTISET(SELECT o.order_id, o.order_mode,
                             MAKE_REF(oc_customers,o.customer_id),
                             o.order_status,
                             o.order_total,o.sales_rep_id,
                             CAST(MULTISET(SELECT l.order_id,l.line_item_id,
                                                    l.unit_price,l.quantity,
                                             MAKE_REF(oc_product_information,
                                                      l.product_id)
                                             FROM order_items l
                                             WHERE o.order_id = l.order_id)
                                    AS order_item_list_typ)
                         FROM orders o
                         WHERE c.customer_id = o.customer_id)
                AS order_list_typ)
FROM customers c;
```

System Privileges

```
SELECT      dbms_metadata.get_granted_ddl('SYSTEM_GRANT') ddl_string
FROM        dual
GRANT QUERY REWRITE TO "OE";
GRANT CREATE MATERIALIZED VIEW TO "OE";
GRANT CREATE DATABASE LINK TO "OE";
GRANT CREATE VIEW TO "OE";
GRANT CREATE SYNONYM TO "OE";
GRANT UNLIMITED TABLESPACE TO "OE";
GRANT CREATE SESSION TO "OE";
```

Retrieve a Single Object and Dependent DDL

```
SELECT     dbms_metadata.get_ddl('TABLE', 'CUSTOMERS') ddl_string
FROM       dual
  CREATE TABLE "OE"."CUSTOMERS"
   (    "CUSTOMER_ID" NUMBER(6,0),
        "CUST_FIRST_NAME" VARCHAR2(20) CONSTRAINT "CUST_FNAME_NN" NOT NULL ENABLE,
        "CUST_LAST_NAME" VARCHAR2(20) CONSTRAINT "CUST_LNAME_NN" NOT NULL ENABLE,
        "CUST_ADDRESS" "OE"."CUST_ADDRESS_TYP" ,
        "PHONE_NUMBERS" "OE"."PHONE_LIST_TYP" ,
        "NLS_LANGUAGE" VARCHAR2(3),
        "NLS_TERRITORY" VARCHAR2(30),
        "CREDIT_LIMIT" NUMBER(9,2),
        "CUST_EMAIL" VARCHAR2(30),
        "ACCOUNT_MGR_ID" NUMBER(6,0),
        "CUST_GEO_LOCATION" "MDSYS"."SDO_GEOMETRY" ,
        "DATE_OF_BIRTH" DATE,
        "MARITAL_STATUS" VARCHAR2(20),
        "GENDER" VARCHAR2(1),
        "INCOME_LEVEL" VARCHAR2(20)
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  TABLESPACE "EXAMPLE" ;
  ALTER TABLE "OE"."CUSTOMERS" ADD CONSTRAINT "CUSTOMER_CREDIT_LIMIT_MAX" CHECK
                  (credit_limit <= 500
0) ENABLE;
  ALTER TABLE "OE"."CUSTOMERS" ADD CONSTRAINT "CUSTOMER_ID_MIN" CHECK (customer_id > 0)
                  ENABLE;
  ALTER TABLE "OE"."CUSTOMERS" ADD CONSTRAINT "CUSTOMERS_PK" PRIMARY KEY ("CUSTOMER_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  TABLESPACE "EXAMPLE"  ENABLE;
  ALTER TABLE "OE"."CUSTOMERS" ADD CONSTRAINT "CUSTOMERS_ACCOUNT_MANAGER_FK" FOREIGN KEY
                  ("ACCOUNT_M
GR_ID")
          REFERENCES "HR"."EMPLOYEES" ("EMPLOYEE_ID") ON DELETE SET NULL ENABLE;
SELECT     dbms_metadata.get_dependent_ddl('OBJECT_GRANT', 'CUSTOMERS')  ddl_string
FROM       dual
  GRANT SELECT ON "OE"."CUSTOMERS" TO "PM";
  GRANT SELECT ON "OE"."CUSTOMERS" TO "BI";
```

Retrieve All Objects of a Type

Extract all the tables owned by OE. At this point, I will disable referential integrity constraints as the order of the tables being output is not guaranteed. The key here is to query user_tables to generate the list of tables.

Bug – cannot deal with nested table.

```
SELECT      DBMS_METADATA.GET_DDL('TABLE', t.table_name) ddl_string
FROM        user_tables t
  CREATE TABLE "HR"."EMPLOYEES"
   (    "EMPLOYEE_ID" NUMBER(6,0),
        "FIRST_NAME" VARCHAR2(20),
        "LAST_NAME" VARCHAR2(25) CONSTRAINT "EMP_LAST_NAME_NN" NOT NULL ENABLE,
        "EMAIL" VARCHAR2(25) CONSTRAINT "EMP_EMAIL_NN" NOT NULL ENABLE,
        "PHONE_NUMBER" VARCHAR2(20),
        "HIRE_DATE" DATE CONSTRAINT "EMP_HIRE_DATE_NN" NOT NULL ENABLE,
        "JOB_ID" VARCHAR2(10) CONSTRAINT "EMP_JOB_NN" NOT NULL ENABLE,
        "SALARY" NUMBER(8,2),
        "COMMISSION_PCT" NUMBER(2,2),
        "MANAGER_ID" NUMBER(6,0),
        "DEPARTMENT_ID" NUMBER(4,0),
         CONSTRAINT "EMP_SALARY_MIN" CHECK (salary > 0) ENABLE,
         CONSTRAINT "EMP_EMAIL_UK" UNIQUE ("EMAIL")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  TABLESPACE "EXAMPLE"  ENABLE,
         CONSTRAINT "EMP_EMP_ID_PK" PRIMARY KEY ("EMPLOYEE_ID")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  TABLESPACE "EXAMPLE"  ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  TABLESPACE "EXAMPLE" ;
  CREATE TABLE "HR"."JOB_HISTORY"
   (    "EMPLOYEE_ID" NUMBER(6,0) CONSTRAINT "JHIST_EMPLOYEE_NN" NOT NULL ENABLE,
        "START_DATE" DATE CONSTRAINT "JHIST_START_DATE_NN" NOT NULL ENABLE,
        "END_DATE" DATE CONSTRAINT "JHIST_END_DATE_NN" NOT NULL ENABLE,
        "JOB_ID" VARCHAR2(10) CONSTRAINT "JHIST_JOB_NN" NOT NULL ENABLE,
        "DEPARTMENT_ID" NUMBER(4,0),
         CONSTRAINT "JHIST_DATE_INTERVAL" CHECK (end_date > start_date) ENABLE,
         CONSTRAINT "JHIST_EMP_ID_ST_DATE_PK" PRIMARY KEY ("EMPLOYEE_ID", "START_DATE")
  USING INDEX PCTFREE 10 INITRANS 2 MAXTRANS 255 COMPUTE STATISTICS
  TABLESPACE "EXAMPLE"  ENABLE
   ) PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255 NOCOMPRESS LOGGING
  TABLESPACE "EXAMPLE" ;
```

Retrieve a Schema's Objects

The documentation mentions being able to do 'exports'. For example, being able to make a simple call to extract all of the ddl for a given schema. Although this was mentioned as being able to be performed in 9i, I did not find a way that worked, so I created a script (case study). In 10g, there is a schema_export function, but I don't think it works properly. It includes statistic insert statements and does not include create user/role statements, so it does not do everything it needs to recreate the user.

CASE STUDY (LESSONS LEARNED)

As a person who learns by example, I like case studies complete with scripts and output. What follows is the actual issue I worked that first acquainted me with dbms_metadata. As I have noted, it was both a frustrating and educational experience.

Project Team Has Lost DDL

One of the project teams we supported needed to redeploy their application, including a new database, to a new server. Unfortunately, they could not find their ddl in source code repository. In discussions with the project team and my own dba team, we realize that the problem was probably not isolated.

They needed a simple, basic, command line tool (no Windows products allowed...officially). One options was to use export with rows=no to create a text file that could be hand edited, but the hand editing was considered to be too time consuming and prone to error.

Challenges

I encountered quite a few challenges over the next few weeks. In some cases there were bugs that were pretty easy to work around, in some cases there was “Expected behavior” that was very problematic.

Bugs

- Malformed SYNONYM calls. *Appears to be fixed in 10gR2.
- VIEW ddl would split words *Appears to be fixed in 10gR2
- TABLESPACE QUOTAS would error even when they existed

“Expected Behaviors”

- If a user did not have a privilege, the call would result in an error. Still in 10gR2
- SQL*Plus cannot differentiate between ‘expected’ and ‘unexpected’ errors.

View Extraction

- Views were stored without spaces between columns SELECT “A1”.”COLUMN1”,”A1”.”COLUMN2”
- No spaces for word breaks *Appears to be fixed in 10gR2
- View text stored in LONG Cannot be manipulated in sql*plus Saved as a CLOB, then manipulated

EXTRACTING SCHEMA DDL

Scripts

All scripts are dependent on SQL*Plus. Feel free to download and adapt them for other tools. The usual caveats apply...especially when dealing with DBMS_METADATA! These scripts must be run with a DBA privileged account (this is checked by the scripts)

extract_schema_menu.sql

This script drives the other two. It has looping and conditional logic built in.

extract_schema_dependencies.sql

This script is called by the menu script to display dependencies. It accepts 1 parameter, the name of the schema owner. This can be used to determine any external dependencies (profiles, tablespaces) that must exist before recreating the schema.

extract_schema_ddl.sql

This script performs the actual extract. It accepts 1 parameter, the name of the schema owner. It will generate the object ddl in proper order. This is important to remember when making additions/changes to the script.

About the Author

Daniel Fink is an independent consultant specializing in Oracle performance diagnosis and optimization, monitoring, training and data recovery. He has been in IT for over 17 years, first as a Cobol programmer on the Vax VMS platform. In 1995, he transitioned to Oracle, first as a developer, then dba…on DEC Alpha OpenVMS and Oracle Parallel Server 7.0.13. He has been both a development and production dba (often times at the same time)

For the past 5 years, Daniel has focused on Oracle performance diagnosis and optimization. He has been a member of the Oak Table Network (www.oaktable.net) since 2004. His Oracle affiliations include the BAARF (Battle Against Any Raid Five) and BAAG (Battle Against Any Guess) Party memberships.

As a partner with Miracle A/S (Denmark), Daniel offers specialized Oracle data recovery using DUDE (Data Unloader through Data Extraction), a platform independent tool that is able to retrieve data from a down and unrecoverable Oracle database.

Daniel has presented at conferences and user group meetings across the United States and Europe, including RMOUG Training Days, The Hotsos Symposium, IOUG-A Live!, UKOUG, Miracle dbForums in Denmark and Scotland. He delivers training in DBA and Development topics, including “A Comprehensive Approach to Performance Diagnosis”, “Effective Performance for Developers and Users” and “Advanced SQL and SQL*Plus”.

Papers, scripts and weblog

Web	http://www.optimaldba.com
Weblog	http://optimaldba.blogspot.com

Contact Information

Phone	+1 303 808 3282
Email	daniel.fink@optimaldba.com