Everything You Always Wanted to Know About Oracle's Data Dictionary

…BUT WERE AFRAID TO ASK!

DAN STOBER
INTERMOUNTAIN HEALTHCARE
EDW TEAM
THURSDAY MARCH 14, 2013

ABOUT ME

- Dan Stober
- Intermountain Healthcare – Since 1995
- Board Member – UTOUG 2010-2012
- California State University, Fresno – Go Bulldogs!
- Contact info: dan.stober@imail.org

SESSION NORMS

- Cell Phones, Pagers, Semaphores – OK
- The slides will be on the UTOUG website
- Questions / comments – Interrupt me!

Agenda

- Why use the data dictionary?
- The data dictionary hierarchy
- Some useful data dictionary views
- Scripting / Building DDL

Why use the Data Dictionary?

- Permits querying of data structures
- Build other SQL statements
  - Including DML and DDL
- Very useful when trying to understand or reverse-engineer a database
  - Which tables join to which tables?
  - How do records get put into this table?
  - What is the relationship between fields in this table?
- Sometimes, Schema Browser is better, sometimes data dictionary is better
  - Schema Browser is just an app that queries the data dictionary

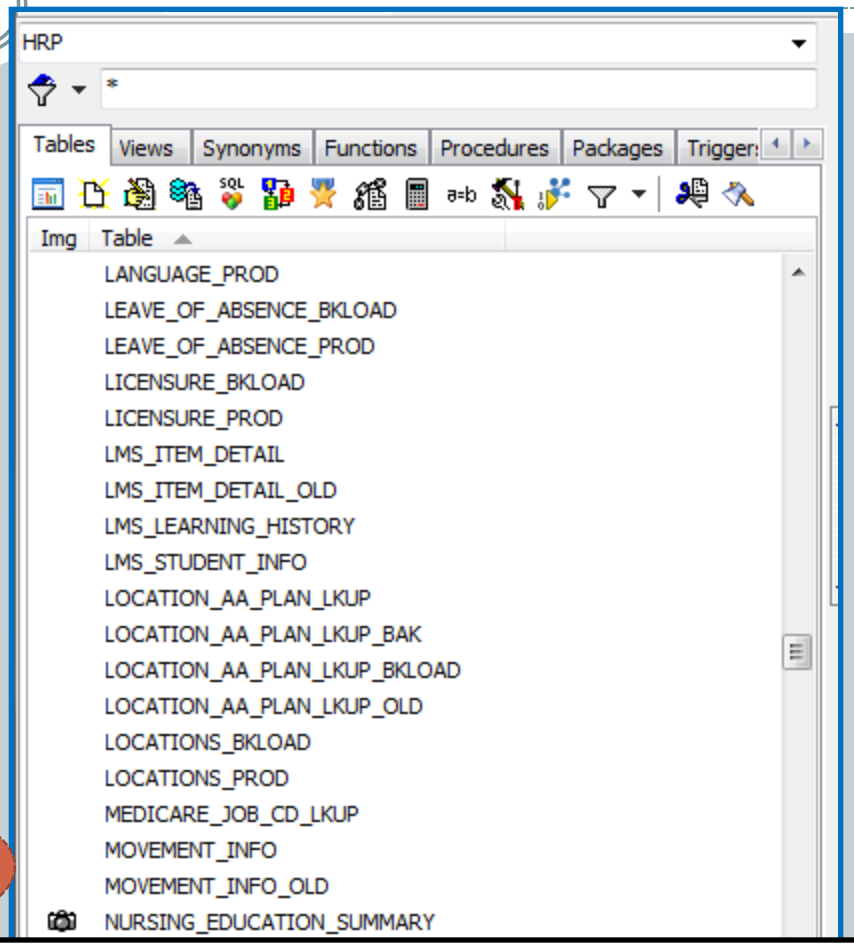
FINDING AN OBJECT

Do you know of a table containing the addresses of all of our hospitals and clinics?

Hmmm, I know we have one.

...is it a table or a view? Or, is it a materialized view?

I think that it has "LOCATION" in the name




```
SELECT * FROM all_objects
WHERE owner = 'HRP'
  AND object_name LIKE '%LOCATION%'
  AND object_type IN ( 'TABLE','VIEW');
```

ALL_OBJECTS

- The records in ALL_OBJECTS explode to further detail in other views

all_tables

all_views

all_mviews

TABLE
VIEW
MATERIALIZED VIEW
FUNCTION
PROCEDURE
PACKAGE
PACKAGE BODY
INDEX
TABLE PARTITION

all_indexes

all_source

THE HIERARCHY

QUERIES TO TEST ACCESS

Connecting as three different users

```
SQL> connect hrp@edw
Password *********

Connected.

SQL> SELECT COUNT(*)
  2    FROM assignment_dim
  3    WHERE ROWNUM = 1;

COUNT(*)
--------
       1

1 row selected.
```

```
SQL> connect emptime@edw
Password *********

Connected.

SQL> SELECT COUNT(*)
  2    FROM assignment_dim
  3    WHERE ROWNUM = 1;

ORA-00904: Table or view does
not exist

SQL> SELECT COUNT(*)
  2    FROM hrp.assignment_dim
  3    WHERE ROWNUM = 1;

COUNT(*)
--------
       1

1 row selected.
```

```
SQL> connect rco@edw
Password *********

Connected.

SQL> SELECT COUNT(*)
  2    FROM assignment_dim
  3    WHERE ROWNUM = 1;

ORA-00904: Table or view does
not exist

SQL> SELECT COUNT(*)
  2    FROM hrp.assignment_dim
  3    WHERE ROWNUM = 1;

ORA-00904: Table or view does
not exist
```

DATA DICTIONARY ENTRIES

connect hrp

```
SQL> SELECT view_name
  2    FROM user_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

VIEW_NAME
-----------------
ASSIGNMENT_DIM

1 row selected.

SQL> SELECT owner, view_name
  2    FROM all_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

OWNER VIEW_NAME
----- ---------------------------
HRP   ASSIGNMENT_DIM

1 row selected.
```

connect emptime

```
SQL> SELECT view_name
  2    FROM user_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

No rows selected.

SQL> SELECT owner, view_name
  2    FROM all_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

OWNER VIEW_NAME
----- ---------------------------
HRP   ASSIGNMENT_DIM

1 row selected.
```

all_views contains an additional column which is not found in **user_views: owner**

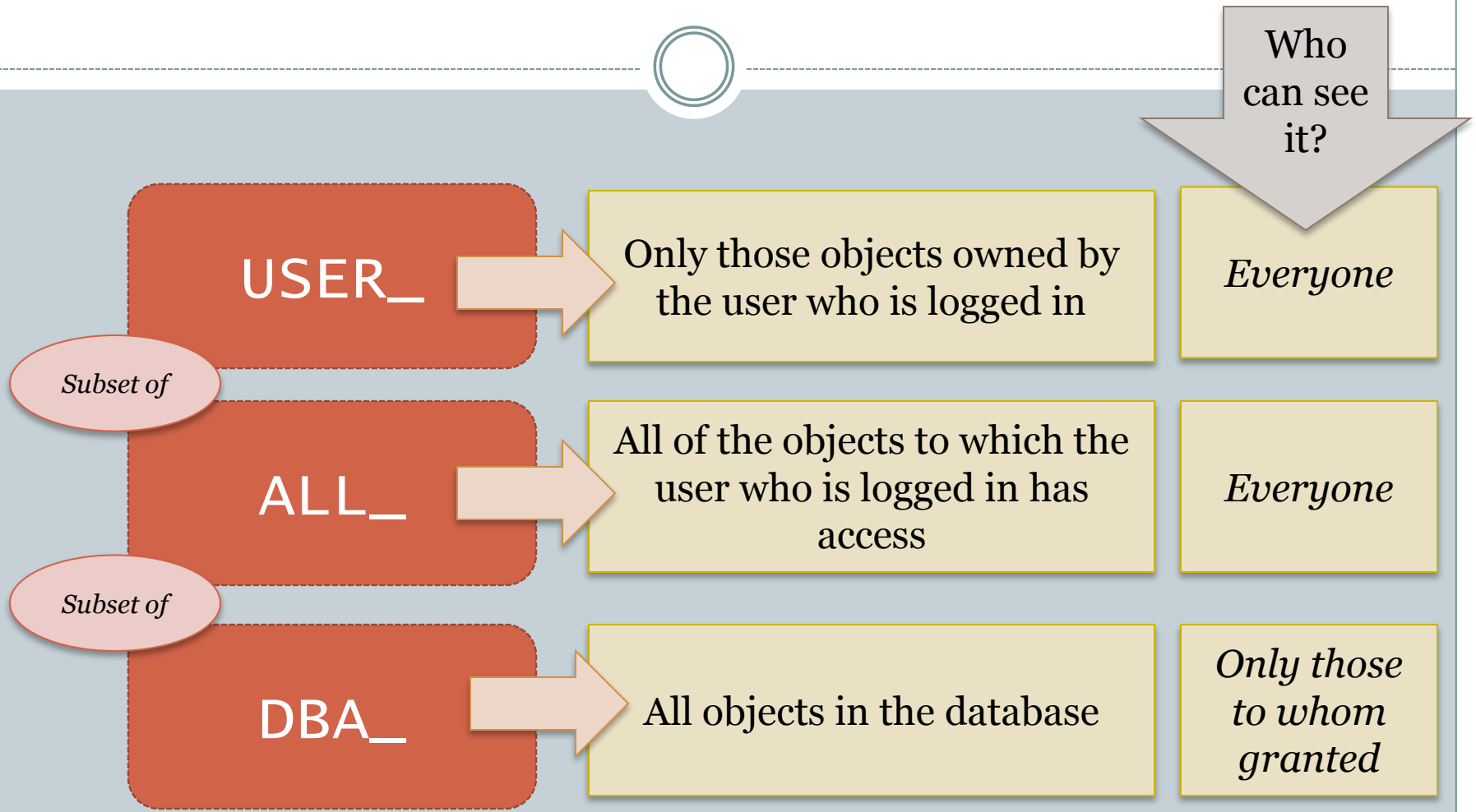
DATA DICTIONARY ENTRIES

connect rco

```
SQL> SELECT view_name
  2    FROM user_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

No rows selected.

SQL> SELECT owner, view_name
  2    FROM all_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

No rows selected.

SQL> SELECT owner, view_name
  2    FROM dba_views
  3    WHERE view_name = 'ASSIGNMENT_DIM';

OWNER VIEW_NAME
----- ---------------------------
HRP   ASSIGNMENT_DIM

1 row selected.
```

Not everyone has access to dba_views

DATA DICTIONARY NAMING SCHEME



THE DATA DICTIONARY VIEWS

ALL_OBJECTS

- The records in ALL_OBJECTS explode to further detail in other views

all_tables

all_views

all_mviews

TABLE
VIEW
MATERIALIZED VIEW
FUNCTION
PROCEDURE
PACKAGE
PACKAGE BODY
INDEX
TABLE PARTITION

all_indexes

all_source

ALL OBJECTS

- All objects in the database

TABLE
VIEW
MATERIALIZED VIEW
FUNCTION
PROCEDURE
PACKAGE
PACKAGE BODY
INDEX
TABLE PARTITION

```
SELECT object_type, COUNT(*)
FROM all_objects
GROUP BY object_type
```

```
DESC all_objects
 Name                 Null?    Type
 -------------------- -------- --------------
 OWNER                         VARCHAR2(30)
 OBJECT_NAME                   VARCHAR2(128)
 SUBOBJECT_NAME                VARCHAR2(30)
 OBJECT_ID                     NUMBER
 DATA_OBJECT_ID                NUMBER
 OBJECT_TYPE                   VARCHAR2(19)
 CREATED                       DATE
 LAST_DDL_TIME                 DATE
 TIMESTAMP                     VARCHAR2(19)
 STATUS                        VARCHAR2(7)
 TEMPORARY                     VARCHAR2(1)
 GENERATED                     VARCHAR2(1)
 SECONDARY                     VARCHAR2(1)
 NAMESPACE                     NUMBER
 EDITION_NAME                  VARCHAR2(30)
```

ALL_TABLES

- Every table* to which the user has access

```
DESC all_tables
 Name                          Null?    Type
 ----------------------------- -------- --------------
 OWNER                         NOT NULL VARCHAR2(30)
 TABLE_NAME                    NOT NULL VARCHAR2(30)
 TABLESPACE_NAME                        VARCHAR2(30)
 CLUSTER_NAME                           VARCHAR2(30)
 IOT_NAME                               VARCHAR2(30)
 STATUS                                 VARCHAR2(8)
 PCT_FREE                               NUMBER
 PCT_USED                               NUMBER
 INI_TRANS                              NUMBER
 MAX_TRANS                              NUMBER
 INITIAL_EXTENT                         NUMBER
 NEXT_EXTENT                            NUMBER
 MIN_EXTENTS                            NUMBER
 MAX_EXTENTS                            NUMBER
 PCT_INCREASE                           NUMBER
 FREELISTS                              NUMBER
 FREELIST_GROUPS                        NUMBER
 LOGGING                                VARCHAR2(3)
 BACKED_UP                              VARCHAR2(1)
 NUM_ROWS                               NUMBER
 BLOCKS                                 NUMBER
 EMPTY_BLOCKS                           NUMBER
 AVG_SPACE                              NUMBER
```

ALL_TABLES

- Back to the “Location” example…

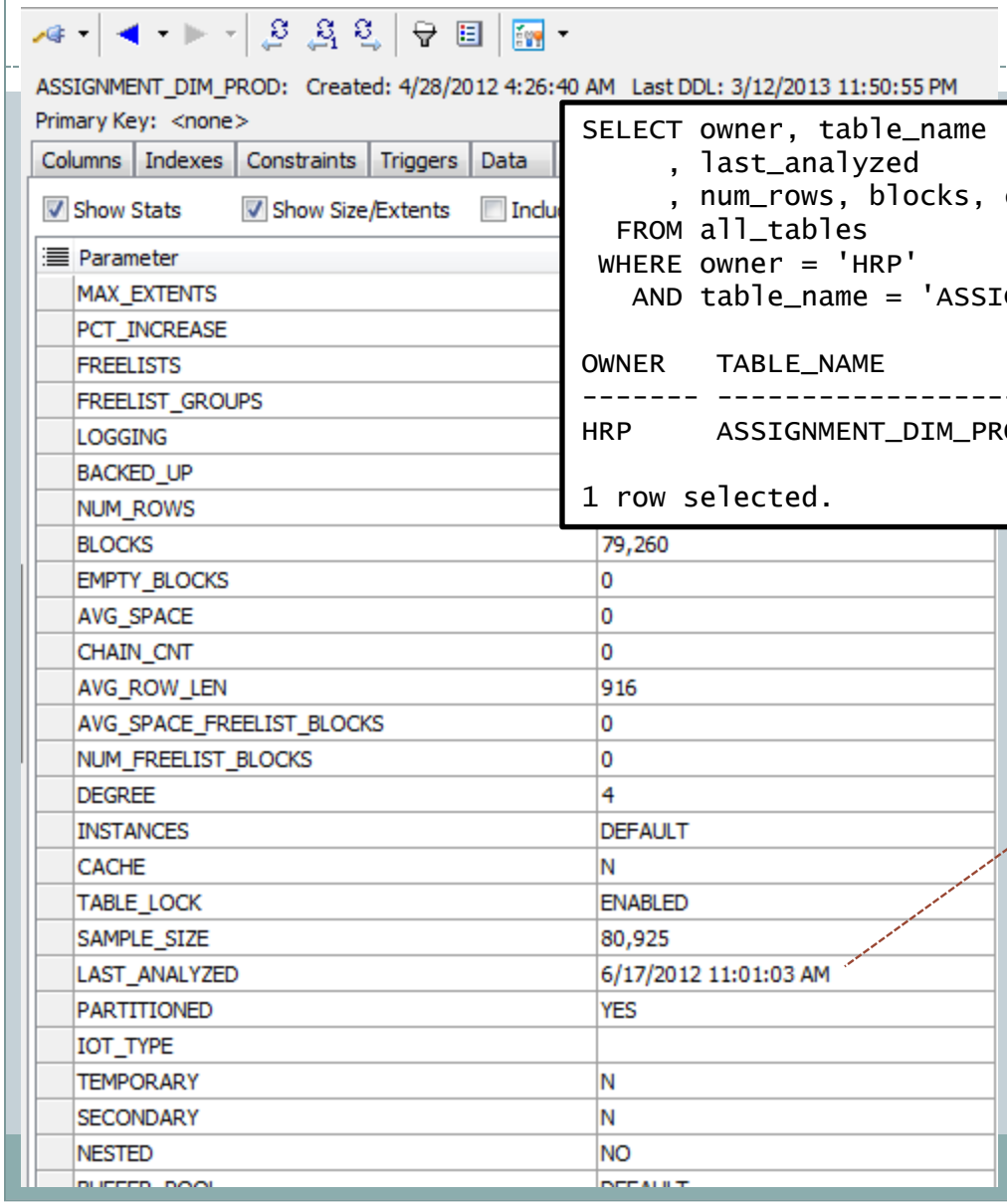
```
SELECT owner, table_name
  FROM all_tables
 WHERE table_name LIKE '%LOC%';

OWNER      TABLE_NAME
---------- ------------------------------
HRP        AAP_LOCATION_BKLOAD
HRP        AAP_LOCATION_PROD
HRP        DIM_LOCATION
HRP        FCILTY_CLINIC_LOCATIONS
HRP        FCILTY_CLINIC_LOCATIONS_BKLOAD
HRP        LOCATIONS_BKLOAD
HRP        LOCATIONS_PROD
HRP        LOCATION_AA_PLAN_LKUP
HRP        LOCATION_AA_PLAN_LKUP_BAK
HRP        LOCATION_AA_PLAN_LKUP_BKLOAD
HRP        LOCATION_AA_PLAN_LKUP_OLD
HRP_STAGE  STAGE_LOCATIONS
HRP_STAGE  STAGE_LOCATION_AAP
HRP_STAGE  T_LOCATIONS
TRAIN      MMS_LOCATION

15 rows selected.
```

@findtab

```
SELECT tab.table_name
   ,  tab.owner
FROM all_tables        tab
WHERE UPPER(tab.table_name) LIKE UPPER ('%&like_table_name%')
ORDER BY tab.owner, tab.table_name
```

```
SQL> @findtab
Enter value for like_table_name: loc

TABLE_NAME                     OWNER
------------------------------ ---------
AAP_LOCATION_BKLOAD            HRP
AAP_LOCATION_PROD              HRP
DIM_LOCATION                   HRP
FCILTY_CLINIC_LOCATIONS        HRP
FCILTY_CLINIC_LOCATIONS_BKLOAD HRP
LOCATIONS_BKLOAD               HRP
LOCATIONS_PROD                 HRP
LOCATION_AA_PLAN_LKUP          HRP
LOCATION_AA_PLAN_LKUP_BAK      HRP
LOCATION_AA_PLAN_LKUP_BKLOAD   HRP
LOCATION_AA_PLAN_LKUP_OLD      HRP
STAGE_LOCATIONS                HRP_STAGE
STAGE_LOCATION_AAP             HRP_STAGE
T_LOCATIONS                    HRP_STAGE
MMS_LOCATION                   TRAIN

15 rows selected.
```

ALL_TABLES




```
SELECT owner, table_name
     , last_analyzed
     , num_rows, blocks, empty_blocks
  FROM all_tables
 WHERE owner = 'HRP'
   AND table_name = 'ASSIGNMENT_DIM_PROD';

OWNER    TABLE_NAME             LAST_ANAL    NUM_ROWS      BLOCKS EMPTY_BLOCKS
-------  ---------------------- --------- ---------- ---------- ------------
HRP      ASSIGNMENT_DIM_PROD    17-JUN-12    2477242       79260            0

1 row selected.
```

No stats collection on this table in nearly nine months

How can I find other tables with the same problem?

ALL_VIEWS

- Every view

```
DESC all_views
 Name                          Null?    Type
 ----------------------------- -------- ----------------------------
 OWNER                         NOT NULL VARCHAR2(30)
 VIEW_NAME                     NOT NULL VARCHAR2(30)
 TEXT_LENGTH                            NUMBER
 TEXT                                   LONG
 TYPE_TEXT_LENGTH                       NUMBER
 TYPE_TEXT                              VARCHAR2(4000)
 OID_TEXT_LENGTH                        NUMBER
 OID_TEXT                               VARCHAR2(4000)
 VIEW_TYPE_OWNER                        VARCHAR2(30)
 VIEW_TYPE                              VARCHAR2(30)
 SUPERVIEW_NAME                         VARCHAR2(30)
 EDITIONING_VIEW                        VARCHAR2(1)
 READ_ONLY                              VARCHAR2(1)
```

```
SELECT owner, view_name
  FROM all_views
 WHERE view_name = 'ALL_VIEWS';

OWNER      VIEW_NAME
---------- ------------------------------
SYS        ALL_VIEWS

1 row selected.
```

Materialized Views are unique

- Represented Twice in ALL_OBJECTS

```
SELECT owner, object_name, object_type
  FROM all_objects
 WHERE object_name = 'NURSE_HOURS_MV';

OWNER      OBJECT_NAME        OBJECT_TYPE
---------- ------------------ -------------------
EMPTIME    NURSE_HOURS_MV     MATERIALIZED VIEW
EMPTIME    NURSE_HOURS_MV     TABLE

2 rows selected.
```

- Represented in <u>two</u> other Data Dictionary views

```
SELECT owner, table_name
  FROM all_tables
 WHERE table_name = 'NURSE_HOURS_MV';

OWNER      TABLE_NAME
---------- ------------------------------
EMPTIME    NURSE_HOURS_MV

1 row selected.
```

```
SELECT owner, mview_name
  FROM all_mviews
  WHERE mview_name = 'NURSE_HOURS_MV';

OWNER      MVIEW_NAME
---------- ------------------------------
EMPTIME    NURSE_HOURS_MV

1 row selected.
```

ALL_MVIEWS

- Materialized Views

REFRESH_MODE
COMMIT
DEMAND
NEVER

REFRESH_METHOD
COMPLETE
FAST
FORCE
NEVER

```
DESC all_mviews
 Name                          Null?    Type
 ----------------------------- -------- ------------
 OWNER                         NOT NULL VARCHAR2(30)
 MVIEW_NAME                    NOT NULL VARCHAR2(30)
 CONTAINER_NAME                NOT NULL VARCHAR2(30)
 QUERY                                  LONG
 QUERY_LEN                              NUMBER(38)
 UPDATABLE                              VARCHAR2(1)
 UPDATE_LOG                             VARCHAR2(30)
 MASTER_ROLLBACK_SEG                    VARCHAR2(30)
 MASTER_LINK                            VARCHAR2(128)
 REWRITE_ENABLED                        VARCHAR2(1)
 REWRITE_CAPABILITY                     VARCHAR2(9)
 REFRESH_MODE                           VARCHAR2(6)
 REFRESH_METHOD                         VARCHAR2(8)
 BUILD_MODE                             VARCHAR2(9)
 FAST_REFRESHABLE                       VARCHAR2(18)
 LAST_REFRESH_TYPE                      VARCHAR2(8)
 LAST_REFRESH_DATE                      DATE
 STALENESS                              VARCHAR2(19)
 AFTER_FAST_REFRESH                     VARCHAR2(19)
 UNKNOWN_PREBUILT                       VARCHAR2(1)
 UNKNOWN_PLSQL_FUNC                     VARCHAR2(1)
 UNKNOWN_EXTERNAL_TABLE                 VARCHAR2(1)
 UNKNOWN_CONSIDER_FRESH                 VARCHAR2(1)
 UNKNOWN_IMPORT                         VARCHAR2(1)
 UNKNOWN_TRUSTED_FD                     VARCHAR2(1)
 COMPILE_STATE                          VARCHAR2(19)
 USE_NO_INDEX                           VARCHAR2(1)
 STALE_SINCE                            DATE
 NUM_PCT_TABLES                         NUMBER
 NUM_FRESH_PCT_REGIONS                  NUMBER
 NUM_STALE_PCT_REGIONS                  NUMBER
```

ALL_MVIEWS

- Finding Materialized in need of refresh

```
SELECT owner, mview_name
, refresh_mode, refresh_method
, stale_since, last_refresh_date
 FROM all_mviews
WHERE owner = 'HRP'
  AND staleness = 'STALE';

OWNER   MVIEW_NAME     REFRES REFRESH_ STALE_SIN LAST_REFR
------- -------------- ------ -------- --------- ---------
HRP     PKLST_AGE_BAND DEMAND COMPLETE 13-FEB-13 09-FEB-13
HRP     PKLST_FCILTY   DEMAND COMPLETE 13-FEB-13 09-FEB-13

2 rows selected.
```

ALL_TAB_COLUMNS

- Shows all columns in every table and view

```
DESC all_tab_columns
 Name                          Null?    Type
 ----------------------------- -------- ------------------------------
 OWNER                         NOT NULL VARCHAR2(30)
 TABLE_NAME                    NOT NULL VARCHAR2(30)
 COLUMN_NAME                   NOT NULL VARCHAR2(30)
 DATA_TYPE                              VARCHAR2(106)
 DATA_TYPE_MOD                          VARCHAR2(3)
 DATA_TYPE_OWNER                        VARCHAR2(30)
 DATA_LENGTH                   NOT NULL NUMBER
 DATA_PRECISION                         NUMBER
 DATA_SCALE                             NUMBER
 NULLABLE                               VARCHAR2(1)
 COLUMN_ID                              NUMBER
 DEFAULT_LENGTH                         NUMBER
 DATA_DEFAULT                           LONG
 NUM_DISTINCT                           NUMBER
 LOW_VALUE                              RAW(32)
 HIGH_VALUE                             RAW(32)
 DENSITY                                NUMBER
 NUM_NULLS                              NUMBER
 NUM_BUCKETS                            NUMBER
 LAST_ANALYZED                          DATE
 SAMPLE_SIZE                            NUMBER
 CHARACTER_SET_NAME                     VARCHAR2(44)
```

What Can I Do with ALL_TAB_COLUMNS?

- Find tables containing a column name
  - Use LIKE % too
- Find all of the columns with a specific datatype
  - DATE, CLOB?
- Useful when you are trying to reverse engineer – "Where does this column join?"

```
SELECT table_name
 FROM all_tab_columns
WHERE column_name = 'PARTICIPANT_ENROLLMENT_ID';

TABLE_NAME
------------------------------
BENEFIT_ENROLLMENT
BENEFIT_RATES

2 rows selected.
```

```
SELECT column_name
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name = 'ASSIGNMENT_DIM_PROD'
  AND column_name  LIKE 'SUPERVISOR%'
ORDER BY 1;

COLUMN_NAME
------------------------------
SUPERVISOR_ASSIGNMENT_ID
SUPERVISOR_FLG
SUPERVISOR_NOW_ASG_ID
SUPERVISOR_NOW_ASG_NO
SUPERVISOR_NOW_EMAIL_ADDR_TXT
SUPERVISOR_NOW_EMP_NO
SUPERVISOR_NOW_FIRST_NM
SUPERVISOR_NOW_ID
SUPERVISOR_NOW_LAST_NM
SUPERVISOR_NOW_PERSON_ID
SUPERVISOR_NOW_PREF_NM
SUPERVISOR_THIS_RECORD_ID
SUPERVISOR_THIS_REC_ASG_ID
SUPERVISOR_THIS_REC_ASG_NO
SUPERVISOR_THIS_REC_EMP_NO
SUPERVISOR_THIS_REC_FIRST_NM
SUPERVISOR_THIS_REC_LAST_NM
SUPERVISOR_THIS_REC_PERSON_ID
SUPERVISOR_THIS_REC_PREF_NM

19 rows selected.
```

Columns in one table but not the other

```
SELECT table_name, COUNT(*)
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name IN ( 'HEALTHY_LIVING_REPORTING'
                    , 'HEALTHY_LIVING_REPORTING_BK' )
GROUP BY table_name;

TABLE_NAME                     COUNT(*)
------------------------------ --------
HEALTHY_LIVING_REPORTING             87
HEALTHY_LIVING_REPORTING_BK          85

2 rows selected.
```

These two table are supposed to be identical, but I have two more columns in one table than the other

```
SELECT column_name, data_type
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name = 'HEALTHY_LIVING_REPORTING'
MINUS
SELECT column_name, data_type
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name = 'HEALTHY_LIVING_REPORTING_BK';

COLUMN_NAME                    DATA_TYPE
------------------------------ ------------
HOVER_Q3_COACH                 VARCHAR2
HREF_Q3_EDUC                   VARCHAR2
IMG_Q3_COACH                   VARCHAR2

3 rows selected.
```

3???

```
SELECT column_name, data_type
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name = 'HEALTHY_LIVING_REPORTING_BK'
MINUS
SELECT column_name, data_type
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name = 'HEALTHY_LIVING_REPORTING';

COLUMN_NAME                    DATA_TYPE
------------------------------ ------------
HRA_LOGIN_ID                   VARCHAR2

1 row selected.
```

Ah-ha!

Finding all of the DATE Columns in a Table

```
SELECT column_name
FROM all_tab_columns
WHERE data_type = 'DATE'
  AND table_name = 'PH_CPNT';

COLUMN_NAME
------------------------------
REV_DTE
CREATE_TSTMP
CREATE_DTE
APPRVL_DTE
WAITLIST_REMDR_SENT
LST_UPD_TSTMP

6 rows selected.
```

Modify query using LISTAGG to get a single row of output

```
SELECT LISTAGG ( column_name, ',') WITHIN GROUP ( ORDER BY column_id ) col_list
 FROM all_tab_columns
WHERE data_type = 'DATE'
AND table_name = 'PH_CPNT';

COL_LIST
--------------------------------------------------------------------------------
REV_DTE,CREATE_TSTMP,CREATE_DTE,APPRVL_DTE,WAITLIST_REMDR_SENT,LST_UPD_TSTMP

1 row selected.
```

Use that output in another query

```
SELECT cpnt_id,
REV_DTE,CREATE_TSTMP,CREATE_DTE,APPRVL_DTE,WAITLIST_REMDR_SENT,LST_UPD_TSTMP
FROM plateau.ph_cpnt
WHERE cpnt_id = '10197';

CPNT_ID  REV_DTE              CREATE_TSTMP         CREATE_DTE           APPRVL_DTE           WAITLIST_REMDR_SENT LST_UPD_TSTMP
-------- -------------------- -------------------- -------------------- -------------------- ------------------- --------------------
10197    10/14/2011 10:05:00 10/14/2011 10:05:53 10/11/2011 00:00:00 04/30/2011 00:00:00                     10/14/2011 10:05:53
10197    10/14/2011 10:05:00 11/10/2012 03:09:34 10/11/2011 00:00:00 04/30/2011 00:00:00                     11/10/2012 03:09:32
10197    10/14/2011 10:05:00 11/10/2012 13:01:45 10/11/2011 00:00:00 04/30/2011 00:00:00                     11/10/2012 03:09:32
10197    10/14/2011 10:05:00 11/10/2012 13:15:22 10/11/2011 00:00:00 04/30/2011 00:00:00                     11/10/2012 03:09:32

4 rows selected.
```

ALL_TAB_COLUMNS

- Another Use Case: Consistency Check

```
SELECT *
FROM (   SELECT column_name, data_type
               , COUNT(*) cnt
               , COUNT(*) OVER ( PARTITION BY column_name ) c1
           FROM all_tab_columns
          WHERE owner = 'HRP'
            AND column_name IN ( 'EMP_NO', 'EMPI', 'DEPT_NO')
          GROUP BY column_name, data_type
     )
WHERE c1 > 1;

COLUMN_NAME                    DATA_TYPE            CNT         C1
------------------------------ ------------ ---------- ----------
DEPT_NO                        NUMBER               36          2
DEPT_NO                        VARCHAR2              3          2
EMPI                           NUMBER                2          2
EMPI                           VARCHAR2              1          2
EMP_NO                         NUMBER                9          2
EMP_NO                         VARCHAR2            105          2

6 rows selected.
```

Recycle Bin Issues

- When tables are dropped, they don't really go away
  - Oracle gives them new names
- They still show up in data_dictionary query results
- To filter out recyclebin tables
- WHERE table_name NOT LIKE 'BIN$%'

```
WHERE table_name NOT LIKE 'BIN$%'
```

ALL_SOURCE / DBA_SOURCE

- Contains all stored PLSQL
- I usually use DBA_SOURCE here
  - If my user doesn't have EXECUTE privileges, then I won't see the procedures

TRIGGER
PACKAGE
FUNCTION
PROCEDURE

```
DESC dba_source
 Name                          Null?    Type
 ----------------------------- -------- ----------------------------
 OWNER                                  VARCHAR2(30)
 NAME                                   VARCHAR2(30)
 TYPE                                   VARCHAR2(12)
 LINE                                   NUMBER
 TEXT                                   VARCHAR2(4000)
```

ALL_SOURCE

```
SELECT line, text
  FROM dba_source
WHERE owner = 'HRP'
  AND name = 'IS_CURRENT_RECORD_TRUNC';

      LINE TEXT
---------- --------------------------------------------------------------------------------
         1 FUNCTION     is_current_record_trunc ( p_start_dt IN DATE , p_end_dt IN DATE )
         2    RETURN NUMBER
         3
         4 DETERMINISTIC
         5   -- Created for use in function-based index
         6   -- Even though this is Boolean logic, return is numeric so it can
         7   --    be used in SQL statements
         8   -- Use this anywhere were you would write TRUNC( SYSDATE ) BETWEEN  x and y
         9
        10   IS
        11
        12   v_ret_val NUMBER;
        13
        14 BEGIN
        15
        16     IF TRUNC( SYSDATE ) BETWEEN p_start_dt AND p_end_dt
        17     THEN
        18
        19         v_ret_val := 1;
        20
        21     ELSE
        22
        23         v_ret_val := 0;
        24
        25     END IF;
        26
        27     RETURN v_ret_val;
        28
        29 END is_current_record_trunc;

29 rows selected
```

Searching in DBA_SOURCE

- How can I find where records get updated in this table?

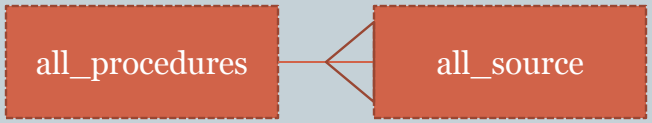
```
SELECT * FROM dba_source
WHERE text LIKE '%audit_historical%';

OWNER      NAME                           TYPE          LINE TEXT
---------- ------------------------------ ----------- ----- ----------------------------------------------------
SANDBOX    KRONOS_EXCEPTIONS              PROCEDURE       12 from emptime.audit_historical ah --
EMPTIME    ETL_HIST_AUDIT_TIMESHEET_ID    PROCEDURE       21   inserted into audit_historical twice in the proc
EMPTIME    TEMP_AUDIT_HIST_NEW_FIELDS     PROCEDURE        4      UPDATE emptime.audit_historical partition) au

3 rows selected.
```

Remember: TEXT is a VARCHAR field; it is case-sensitive!

```
SELECT * FROM dba_source
WHERE LOWER(text) LIKE '%audit_historical%';

OWNER      NAME                           TYPE          LINE TEXT
---------- ------------------------------ ----------- ----- ----------------------------------------------------
SANDBOX    KRONOS_EXCEPTIONS              PROCEDURE       12 from emptime.audit_historical ah --
SANDBOX    MANUAL_PUNCH_GRAPH             PROCEDURE       27      FROM emptime.AUDIT_HISTORICAL, --
EMPTIME    ETL_HISTORICAL_AUDIT           PROCEDURE       16 This procedure is called in the Audit Historical ETL
EMPTIME    ETL_HIST_AUDIT_TIMESHEET_ID    PROCEDURE       21   inserted into audit_historical twice in the proc
EMPTIME    ETL_HISTORICAL                 PROCEDURE      115  ( table_nm => 'AUDIT_HISTORICAL' );
EMPTIME    ETL_HISTORICAL                 PROCEDURE      134    edw_util.stats_table('AUDIT_HISTORICAL');
EMPTIME    TEMP_AUDIT_HIST_NEW_FIELDS     PROCEDURE        4      UPDATE emptime.audit_historical partition) au

7 rows selected.
```

ALL_PROCEDURES

- Various object types
- NULL procedure_name
- Overload
- Determinstic



- TRIGGER
- PACKAGE
- FUNCTION
- PROCEDURE
- TYPE

```
SELECT owner, object_name, procedure_name
 , overload, object_type, deterministic
 FROM dba_procedures
 WHERE owner = 'HRP'
 AND object_name IN ( 'METADATA_PKG','ETL_ASSIGNMENT_CHANGES','IS_CURRENT_RECORD_TRUNC');

OWNER      OBJECT_NAME                    PROCEDURE_NAME                 OVERLOAD OBJECT_TYPE    DET
---------- ------------------------------ ------------------------------ -------- ------------- ---
HRP        ETL_ASSIGNMENT_CHANGES                                                 PROCEDURE     NO
HRP        IS_CURRENT_RECORD_TRUNC                                                FUNCTION      YES
HRP        METADATA_PKG                   COPY_ALL_COLUMN_COMMENTS                PACKAGE       NO
HRP        METADATA_PKG                   COPY_SINGLE_COLUMN_COMMENTS             PACKAGE       NO
HRP        METADATA_PKG                   GET_VPD_POLICY                 1        PACKAGE       NO
HRP        METADATA_PKG                   GET_VPD_POLICY                 2        PACKAGE       NO
HRP        METADATA_PKG                   SINGLE_TABLE_VIEWS_COMMENTS             PACKAGE       NO
HRP        METADATA_PKG                   TABLE_SWAP_COMMENTS                     PACKAGE       NO
HRP        METADATA_PKG                   UPDATE_METADATA_REVIEW_DATE             PACKAGE       NO
HRP        METADATA_PKG                                                           PACKAGE       NO

10 rows selected.
```

ALL_DEPENDENCIES

- Shows objects which are dependent on others

```
DESC all_dependencies
 Name                                      Null?    Type
 ----------------------------------------- -------- ---------------
 OWNER                                     NOT NULL VARCHAR2(30)
 NAME                                      NOT NULL VARCHAR2(30)
 TYPE                                               VARCHAR2(18)
 REFERENCED_OWNER                                   VARCHAR2(30)
 REFERENCED_NAME                                    VARCHAR2(64)
 REFERENCED_TYPE                                    VARCHAR2(18)
 REFERENCED_LINK_NAME                               VARCHAR2(128)
 DEPENDENCY_TYPE                                    VARCHAR2(4)
```

```
SELECT owner, name
   , referenced_owner, referenced_name
FROM all_dependencies
WHERE referenced_name = 'AUDIT_HISTORICAL';

OWNER      NAME                           REFERENCED_OWNER               REFERENCED_NAME
---------- ------------------------------ ------------------------------ ----------------
EMPTIME    KRONOS_EXCEPTION               EMPTIME                        AUDIT_HISTORICAL

1 row selected.
```

Comments

- Table comments and column comments

all_tab_comments

```
DESC all_tab_comments
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                     NOT NULL VARCHAR2(30)
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 TABLE_TYPE                                         VARCHAR2(11)
 COMMENTS                                           VARCHAR2(4000)
```

all_tab_comments

```
DESC all_col_comments
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OWNER                                     NOT NULL VARCHAR2(30)
 TABLE_NAME                                NOT NULL VARCHAR2(30)
 COLUMN_NAME                               NOT NULL VARCHAR2(30)
 COMMENTS                                           VARCHAR2(4000)
```

VPD / Row Level Security

- Table comments and column comments

dba_policies

```
DESC dba_policies
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OBJECT_OWNER                              NOT NULL VARCHAR2(30)
 OBJECT_NAME                               NOT NULL VARCHAR2(30)
 POLICY_GROUP                              NOT NULL VARCHAR2(30)
 POLICY_NAME                               NOT NULL VARCHAR2(30)
 PF_OWNER                                  NOT NULL VARCHAR2(30)
 PACKAGE                                            VARCHAR2(30)
 FUNCTION                                  NOT NULL VARCHAR2(30)
 SEL                                                VARCHAR2(3)
 INS                                                VARCHAR2(3)
 UPD                                                VARCHAR2(3)
 DEL                                                VARCHAR2(3)
 IDX                                                VARCHAR2(3)
                                                    VARCHAR2(3)
                                                    VARCHAR2(3)
```

dba_sec_relevant_cols

```
DESC dba_sec_relevant_cols
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 OBJECT_OWNER                              NOT NULL VARCHAR2(30)
 OBJECT_NAME                               NOT NULL VARCHAR2(30)
 POLICY_GROUP                              NOT NULL VARCHAR2(30)
 POLICY_NAME                               NOT NULL VARCHAR2(30)
 SEC_REL_COLUMN                            NOT NULL VARCHAR2(30)
 COLUMN_OPTION                                      VARCHAR2(8)
```

ALL_ERRORS

- Creating view
  - Errors
  - Show errors
  - No errors?
- Warning: View created with compilation errors.
- select text from user_errors
- where name = 'ASSIGNMENT_DIM'

ALL_ERRORS

```
*

ERROR at line 1:
ORA-04063: package body "IDX.LOAD_PKG" has errors
ORA-06508: PL/SQL: could not find program unit being called: "IDX.LOAD_PKG"
ORA-06512: at "IDX.LOAD_SCHPROV", line 95
ORA-06512: at line 1
```

```
SELECT * FROM user_errors
WHERE name = 'LOAD_PKG';

NAME     TYPE           SEQUENCE       LINE   POSITION TEXT
-------- ------------ ---------- ---------- ---------- ------------------------------------------------
LOAD_PKG PACKAGE BODY          1        477         26 PL/SQL: ORA-00942: table or view does not exist
LOAD_PKG PACKAGE BODY          2        476          6 PL/SQL: SQL Statement ignored
LOAD_PKG PACKAGE BODY          3        483         22 PL/SQL: ORA-00942: table or view does not exist
LOAD_PKG PACKAGE BODY          4        483          6 PL/SQL: SQL Statement ignored

4 rows selected.
```

ALL_TAB_PARTITIONS

- Shows all partitions from partitioned tables
- Another one for subpartitions, too

```
DESC all_tab_partitions
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 TABLE_OWNER                                        VARCHAR2(30)
 TABLE_NAME                                         VARCHAR2(30)
 COMPOSITE                                          VARCHAR2(3)
 PARTITION_NAME                                     VARCHAR2(30)
 SUBPARTITION_COUNT                                 NUMBER
 HIGH_VALUE                                         LONG
 HIGH_VALUE_LENGTH                                  NUMBER
 PARTITION_POSITION                                 NUMBER
 TABLESPACE_NAME                                    VARCHAR2(30)
 PCT_FREE                                           NUMBER
 PCT_USED                                           NUMBER
 INI_TRANS                                          NUMBER
```

TOAD

```
15  SELECT table_owner, table_name
16  , partition_name, high_value
17  FROM all_tab_partitions
18  WHERE table_owner = 'IDX'
19    AND table_name = 'LINE_ITEM';
```

Data Grid

Messages | Data Grid | Trace | DBMS Output (disabled)

TABLE_OWNER	TABLE_NAME	PARTITION_NAME	HIGH_VALUE
IDX	LINE_ITEM	LASTONE	(WIDEMEMO)
IDX	LINE_ITEM	MO00	(WIDEMEMO)
IDX	LINE_ITEM	MO03	(WIDEMEMO)
IDX	LINE_ITEM	MO06	(WIDEMEMO)
IDX	LINE_ITEM	MO09	(WIDEMEMO)
IDX	LINE_ITEM	MO12	(WIDEMEMO)
IDX	LINE_ITEM	MO15	(WIDEMEMO)
IDX	LINE_ITEM	MO18	(WIDEMEMO)

SQL DEVELOPER

```
SELECT table_owner, table_name
  , partition_name, high_value
FROM all_tab_partitions
WHERE table_owner = 'IDX'
  AND table_name = 'LINE_ITEM';
```

Script Output | Explain Plan | Query Result

SQL | All Rows Fetched: 15 in 0.738 seconds

	TABLE_OWNER	TABLE_NAME	PARTITION_NAME	HIGH_VALUE
1	IDX	LINE_ITEM	LASTONE	MAXVALUE
2	IDX	LINE_ITEM	MO00	TO_DATE(' 2013-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
3	IDX	LINE_ITEM	MO03	TO_DATE(' 2012-11-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
4	IDX	LINE_ITEM	MO06	TO_DATE(' 2012-08-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
5	IDX	LINE_ITEM	MO09	TO_DATE(' 2012-05-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
6	IDX	LINE_ITEM	MO12	TO_DATE(' 2012-02-01 00:00:00', 'SYYYY-MM-DD HH24:MI:SS', 'NLS_CALENDAR=GREGORIAN')
7	IDX	LINE_ITEM	MO15	TO_DATE(' 2011-11
8	IDX	LINE_ITEM	MO18	TO_DATE(' 2011-08

SQL*Plus will display the LONG column in readable form, too

Querying Partitions

```
SELECT table_owner, table_name
  , partition_name, high_value
 FROM all_tab_partitions
WHERE table_owner = 'IDX'
  AND table_name = 'LINE_ITEM'
  AND high_value < DATE '2011-01-01';

, partition_name, high_value
                  *
ERROR at line 2:
ORA-00997: illegal use of LONG datatype
```

Custom function

```
SELECT table_owner, table_name
, partition_name
, idx.load_pkg.get_range_partition_date ( table_owner, table_name, partition_name )
 FROM all_tab_partitions
 WHERE table_owner = 'IDX'
   AND table_name = 'LINE_ITEM'
   AND idx.load_pkg.get_range_partition_date
              ( table_owner, table_name, partition_name ) < DATE '2011-01-01'
11:04:08 SQL> /

TABLE_OWNER TABLE_NAME                     PARTITION_NAME                 IDX.LOAD_
----------- ------------------------------ ------------------------------ ---------
IDX         LINE_ITEM                      YR03                           01-FEB-10
IDX         LINE_ITEM                      YR04                           01-FEB-09
IDX         LINE_ITEM                      YR05                           01-FEB-08
```

Getting Partition Date Ranges

```
  FUNCTION get_range_partition_date
      (   p_table_owner     IN VARCHAR2
      ,   p_table_name      IN VARCHAR2
      ,   p_partition_name  IN VARCHAR2  )
  RETURN DATE
  IS

     v_sql_txt              VARCHAR2(500);

     v_high_value           VARCHAR2(500);
     v_high_value_length    NUMBER;

     v_ret_val              DATE;

BEGIN

    BEGIN

   SELECT  high_value
        ,  high_value_length
     INTO  v_high_value
        ,  v_high_value_length
     FROM all_tab_partitions
    WHERE  table_owner    = UPPER( p_table_owner )
      AND  table_name     = UPPER( p_table_name  )
      AND  partition_name = UPPER( p_partition_name  );

    EXCEPTION
        WHEN no_data_found THEN

            v_high_value       := NULL;
            v_ret_val          := NULL;

    END;

    IF v_high_value = 'MAXVALUE' THEN

        v_ret_val  := NULL;

    ELSIF v_high_value IS NOT NULL THEN

        v_sql_txt := 'SELECT ' || SUBSTR ( v_high_value, 1, v_high_value_length ) || ' FROM DUAL';

        EXECUTE IMMEDIATE v_sql_txt INTO v_ret_val;

    END IF;

EXCE
    _

   WHEN OTHERS THEN

        RAISE;

END;
```

```
v_ret_val  DATE;
```

```
SELECT high_value
    ,  high_value_length
INTO v_high_value
   , v_high_value_length
FROM all_tab_partitions
WHERE  table_owner    = UPPER( p_table_owner )
  AND  table_name     = UPPER( p_table_name  )
  AND  partition_name = UPPER( p_partition_name );
```

```
v_sql_txt := 'SELECT ' || SUBSTR ( v_high_value, 1, v_high_value_length ) || ' FROM DUAL';

EXECUTE IMMEDIATE v_sql_txt INTO v_ret_val;
```

Complete source code can be found in the comments to this slide

Advanced Partition Quuery

- Finding DATE partitioned tables that do not use automatic interval

```
SELECT
  dpkc.owner, dpkc.name, dpkc.column_name, dtc.data_type, dtp.interval_flg
  FROM dba_part_key_columns dpkc
     JOIN dba_tab_columns dtc ON (  dpkc.owner = dtc.owner
                                AND dpkc.name = dtc.table_name
                                AND dpkc.column_name = dtc.column_name )
     JOIN (SELECT table_owner, table_name
           , MAX(CASE WHEN interval = 'YES' THEN 1 ELSE 0 END) AS interval_flg
             FROM dba_tab_partitions
          GROUP BY table_owner, table_name
          ) dtp  ON (dtp.table_owner = dpkc.owner and dtp.table_name = dpkc.name)
--- unused?
            JOIN metadata.schema s ON (s.schema_name = dpkc.owner)
WHERE
  dpkc.object_type = 'TABLE'
  AND dtc.data_type = 'DATE'
  AND dtp.interval_flg = 0
  AND name NOT LIKE 'BIN$%'
ORDER BY owner, name
```

Courtesy of Steve Catmull

Privileges

- In general, I use USER_ when querying privileges
- `USER_ROLE_PRIVS` – Which roles does the current user have?
- `ALL_TAB_PRIVS` – Two in one!
  - Show Which tables and views does the current user have

  AND…
  - Show are the users and roles who have been granted access to any object owned by the current user?
  - This DOES NOT show objects which can be accessed as the result of a role grant

USER_TAB_PRIVS

- Finding everything that has been granted to my user
  - Includes which privilege (INSERT, UPDATE, etc)
- Notice the EXECUTE privilege
  - This contains more than just tables

```
SELECT * FROM user_tab_privs;

GRANTEE    OWNER      TABLE_NAME                     GRANTOR      PRIVILEGE
---------- ---------- ------------------------------ ----------- -----------
CODSTOBE   IDX        LOAD_PKG                       IDX          EXECUTE
CODSTOBE   SANDBOX    FAIR_PT_TYPES                  SANDBOX      SELECT
CODSTOBE   SANDBOX    LKUP_CAU_STATE                 SANDBOX      SELECT

                                    . . .

CODSTOBE   RCO        EMP                            RCO          DELETE
CODSTOBE   RCO        EMP                            RCO          INSERT
CODSTOBE   RCO        EMP                            RCO          SELECT
CODSTOBE   RCO        EMP                            RCO          UPDATE

51 rows selected.
```

Privileges

```
DESC user_tab_privs
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 GRANTEE                       NOT NULL VARCHAR2(30)
 OWNER                         NOT NULL VARCHAR2(30)
 TABLE_NAME                    NOT NULL VARCHAR2(30)
 GRANTOR                       NOT NULL VARCHAR2(30)
 PRIVILEGE                     NOT NULL VARCHAR2(40)
 GRANTABLE                              VARCHAR2(3)
 HIERARCHY                              VARCHAR2(3)
```

"TABLE_NAME" But it contains more than just tables and views

"TABLE_SCHEMA" instead of owner

```
DESC all_tab_privs
 Name                          Null?    Type
 ----------------------------- -------- --------------------
 GRANTOR                       NOT NULL VARCHAR2(30)
 GRANTEE                       NOT NULL VARCHAR2(30)
 TABLE_SCHEMA                  NOT NULL VARCHAR2(30)
 TABLE_NAME                    NOT NULL VARCHAR2(30)
 PRIVILEGE                     NOT NULL VARCHAR2(40)
 GRANTABLE                              VARCHAR2(3)
 HIERARCHY                              VARCHAR2(3)
```

All_tab_privs shows everything to which you have (direct grant) privileges, PLUS any privileges you've granted

SOME SCRIPTING EXAMPLES

ALL_TAB_COLUMNS: Use Case

HOVER_Q2_COACH	67			Y	VARCHAR2 (100 Byte)
HREF_Q2_COACH	68			Y	VARCHAR2 (40 Byte)
IMG_Q2_REWARD	69			Y	VARCHAR2 (25 Byte)
HOVER_Q2_REWARD	70			Y	VARCHAR2 (100 Byte)
IMG_Q3_EDUC	71			Y	VARCHAR2 (25 Byte)
HOVER_Q3_EDUC	72			Y	VARCHAR2 (100 Byte)
HREF_Q3_EDUC	73			Y	VARCHAR2 (40 Byte)
IMG_Q3_COACH	74			Y	VARCHAR2 (25 Byte)
HOVER_Q3_COACH	75			Y	VARCHAR2 (100 Byte)
	76			Y	VARCHAR2 (40 Byte)
	77			Y	VARCHAR2 (25 Byte)
HOVER_Q3_REWARD					
IMG_Q4_EDUC					
HOVER_Q4_EDUC					
HREF_Q4_EDUC					
IMG_Q4_COACH					
HREF_Q4_COACH					
HOVER_Q4_COACH					

I want to change the size of all of the fields which start with HREF from 40 characters to 100

SOLTUION: Build DDL by concatenating

column_name

table_name

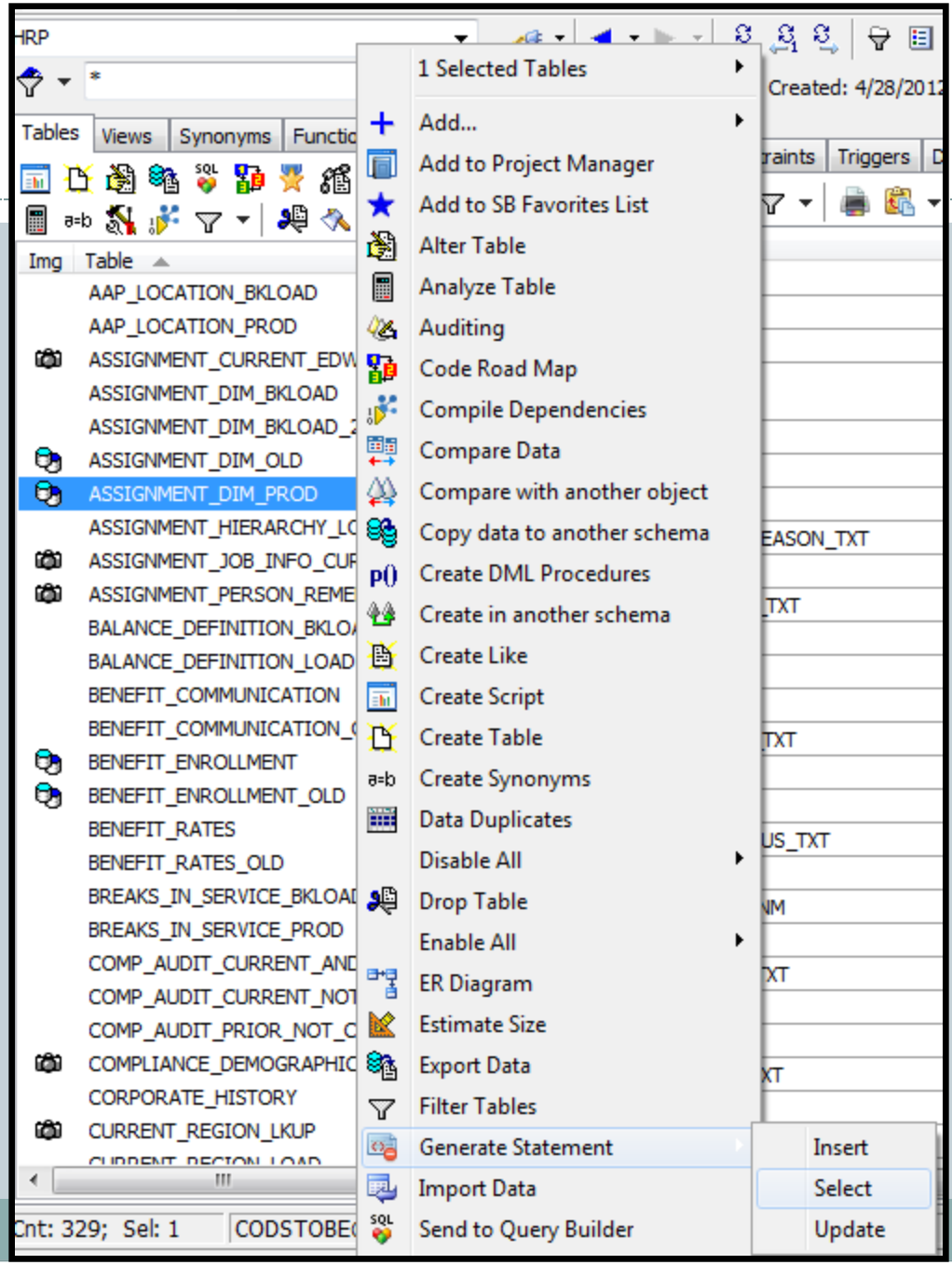
```
SELECT 'ALTER TABLE ' || table_name || ' MODIFY ( '
     || column_name || ' VARCHAR2(100));'
  FROM all_tab_columns
 WHERE owner = 'HRP'
   AND table_name = 'HEALTHY_LIVING_REPORTING'
   AND column_name LIKE 'HREF%';

ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_HRA VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q1_EDUC VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q2_EDUC VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q2_COACH VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q3_EDUC VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q3_COACH VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q4_EDUC VARCHAR2(100));
ALTER TABLE HEALTHY_LIVING_REPORTING MODIFY ( HREF_Q4_COACH VARCHAR2(100));

8 rows selected.
```

BUILDING A SELECT STATEMENT

```
SELECT *
FROM hrp.assignment_dim_prod;
```

```
SELECT a whole bunch of fields…
FROM hrp.assignment_dim_prod;
```

- In TOAD's Schema Browser
  - Right click on table or view name
  - Choose "Generate statement"
  - "Select"



```
SELECT LISTAGG ( column_name, ',') WITHIN GROUP ( ORDER BY column_id ) AS col
FROM all_tab_columns
WHERE owner = 'HRP'
  AND table_name = 'ASSIGNMENT_DIM_PROD';

COL
--------------------------------------------------------------------------------
ASSIGNMENT_KEY,ASSIGNMENT_ID,PERSON_ID,START_DT,END_DT,REC_SEQ_NO,AA_GROUP_NM,ASSIGNMENT_C
HNG_REASON_TXT,ASSIGNMENT_NO,ASSIGNMENT_STATUS_TXT,BENEFITS_ELIG_FLG,CENSUS_CD,CITY_ALLOW_
AMT,CITY_FILING_STATUS_TXT,COMPA_RATIO_VAL,COUNTY_ALLOW_AMT,COUNTY_FILING_STATUS_TXT,EEO_G
ROUP_NM,ASSIGNMENT_CTGRY_NM,FED_ALLOW_AMT,FED_FILING_STATUS_TXT,GL_ACCT_NO,GRADE_LEVEL_TXT
,GRADE_STRUCTURE_TXT,GRADE_TXT,GRE_NM,HIRE_REASON_TXT,HIRING_FORMULA_TXT,HOURLY_RATE_AMT,H
R_PAYROLL_FLG,JOB_CD,JOB_TITLE_TXT,LABOR_MKT_GRP_NM,LAST_SAL_CHNG_PCT,LOC_ADDR_LINE1_TXT,L
OC_CITY_NM,LOC_COUNTRY_CD,LOC_NM,LOC_POSTAL_CD,LOC_STATE_NM,MGT_COUNCIL_TXT,ORG_DEPT_NM,OR
G_NM,PAYROLL_NM,PHYS_EXEC_LTD_ELIG_TXT,PRMRY_ASSIGNMENT_FLG,PTO_GROUP_NM,RANGE_MAX_VAL,RAN
GE_MDPNT_VAL,RANGE_MIN_VAL,REGION_NM,SALARY_BASIS_TXT,SALARY_CHNG_REASON_TXT,SPECIAL_PAY_E
MPL_TXT,STATE_ALLOW_AMT,STATE_FILING_STATUS_TXT,SUB_DEPT_NM,SUPERVISOR_NOW_ID,SUPERVISOR_T
HIS_RECORD_ID,WORKING_HRS,PEOPLE_GROUP_ID,JOB_ID,GRADE_ID,PAYROLL_ID,SOFT_CODING_KEYFLEX_I
D,LOCATION_ID,ORGANIZATION_ID,ASSIGNMENT_CHANGE_FLG,HOURLY_RATE_BAND_TXT,WORKING_HRS_BAND_
TXT,COMPA_RATIO_BAND_TXT,PAYROLL_TENURE_BAND_TXT,DEPT_TENURE_BAND_TXT,MOVEMENT_FLG,EXTENDE
D_END_DT,PAYROLL_TENURE_YRS,DEPT_TENURE_YRS,MOVEMENT_TO_FLG,MOVEMENT_FROM_FLG,JOB_FAMILY_T
XT,CLINICAL_NON_CLINICAL_TXT,JOB_CD_TENURE_BAND,JOB_CD_TENURE_YRS,FCILTY_NO,HR_CONSULTANT_
ID,FCILTY_MOVEMENT_TO_FLG,FCILTY_MOVEMENT_FROM_FLG,FLSA_CD,LAST_RATE_CHANGE_DT,EEO_GROUP_C
D,AA_GROUP_CD,LOCATION_CD,GL_DESCRIPTION_TXT,JOB_FAMILY_SHORT_TXT,REQ_NO,REGION_NOW_NM,SUP
ERVISOR_NOW_FIRST_NM,SUPERVISOR_NOW_LAST_NM,SUPERVISOR_NOW_EMP_NO,SUPERVISOR_NOW_EMAIL_ADD
R_TXT,SUPERVISOR_THIS_REC_FIRST_NM,SUPERVISOR_THIS_REC_LAST_NM,SUPERVISOR_THIS_REC_EMP_NO,
ASSIGNMENT_CTGRY_TEMPORARY_FLG,JOB_COMMENTS_TXT,EDW_LOAD_DT,PAYROLL_START_DT,DEPT_START_DT
,JOB_CD_START_DT,JOB_FAMILY_START_DT,ASSIGNMENT_STATUS_START_DT,LOC_START_DT,WORKING_HRS_S
TART_DT,JOB_FAMILY_TENURE_YRS,RANGE_PENETRATION,HR_CONSULTANT_FIRST_NM,HR_CONSULTANT_LAST_
NM,HR_CONSULTANT_EMP_NO,HR_CONSULTANT_EMAIL_ADDR_TXT,WORKING_HRS_ALL_ASSIGNMENTS,FULL_TIME
_FLG,CURRENT_RECORD_FLG,SUPERVISOR_ASSIGNMENT_ID,CURRENT_SUPERVISOR_FLG,SUPERVISOR_FLG,EMP
_NO,LAST_SAL_CHNG_AMT,SUPERVISOR_NOW_PREF_NM,SUPERVISOR_THIS_REC_PREF_NM,HR_CONSULTANT_PRE
F_NM,SUPERVISOR_THIS_REC_PERSON_ID,SUPERVISOR_NOW_PERSON_ID,SUPERVISOR_THIS_REC_ASG_ID,SUP
ERVISOR_NOW_ASG_ID,DEPT_NO,SUPERVISOR_THIS_REC_ASG_NO,SUPERVISOR_NOW_ASG_NO

1 row selected.
```

Pasting column list into query

```
1
2
3   SELECT
4
5   FROM hrp.assignment_dim_prod;
6
7
```

```
1
2
3   SELECT
4   ASSIGNMENT_KEY,ASSIGNMENT_ID,PERSON_ID,START_DT,END_DT,REC_SEQ_NO,AA_GRO
5   FROM hrp.assignment_dim_prod;
6
7
8
```

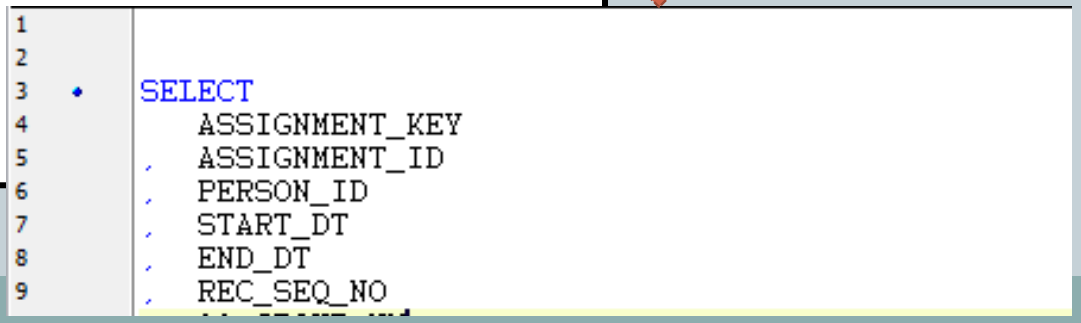
Data Grid

Messages | Data Grid | Explain Plan | Query Viewer | Script Output

ASSIGNMENT_KEY	ASSIGNMENT_ID	PERSON_ID	START_DT	END_DT	REC_SEQ_NO	AA_GROUP_NM
7403885	716	1374	1/1/2013	2/5/2013	59	Technical - Clinical
8305598	24241	59917	1/1/2013	12/31/4712	47	Service Workers (Patient
7434172	2020	4426	1/4/2013	1/19/2013	143	Admin Counsel/VP/Asst V
7437798	2095	4671	1/17/2013	2/14/2013	87	Service Workers (Patient
8302536	24190	59763	1/14/2013	2/3/2013	251	Office & Clerical (Traditio
7442097	2170	4910	1/20/2013	12/31/4712	107	Technical - Clinical
7708870	9983	23130	1/6/2013	12/31/4712	81	Professional - MD/Doctor
8309716	24310	60117	2/20/2013	2/24/2013	656	Managers/Asst Directors
9962170	211239	438523	1/1/2013	5/31/2013	8	Service Workers (Non-Pa

BUILDING A SELECT STATEMENT

```
SELECT DECODE ( ROWNUM, 1, '   ' , ',  ' ) || column_name
  FROM all_tab_columns
 WHERE owner = 'HRP'
   AND table_name = 'ASSIGNMENT_DIM_PROD'
ORDER BY column_id;

DECODE(ROWNUM,1,'',',')||COLUMN_NAME
------------------------------------------------------------
   ASSIGNMENT_KEY
,  ASSIGNMENT_ID
,  PERSON_ID
,  START_DT
,  END_DT
,  REC_SEQ_NO
,  AA_GROUP_NM


                              . . .

,  NEW_BUSINESS_UNIT_CD_BETA
,  NEW_DEPT_CD_BETA
,  NEW_ACCT_CD_BETA

140 rows selected.
```



Constructing DDL from SQL Statement

1. Fine tune the query to make sure the appropriate records are selected

```
SELECT table_name
  FROM user_tables
 WHERE table_name LIKE 'LMS%';

TABLE_NAME
------------------------------
LMS_STUDENT_INFO
LMS_LEARNING_HISTORY
LMS_ITEM_DETAIL_OLD
LMS_ITEM_DETAIL

4 rows selected.

SELECT table_name
  FROM user_tables
 WHERE table_name LIKE 'LMS%'
   AND table_name NOT LIKE '%OLD';

TABLE_NAME
------------------------------
LMS_STUDENT_INFO
LMS_LEARNING_HISTORY
LMS_ITEM_DETAIL

3 rows selected.
```

2. Build the DDL around the data dictionary columns using concatenation

```
SELECT 'GRANT SELECT ON '|| table_name ||' TO lms_select;' cmd
  FROM user_tables
 WHERE table_name LIKE 'LMS%'
   AND table_name NOT LIKE '%OLD';

CMD
-----------------------------------------------------------------
GRANT SELECT ON LMS_STUDENT_INFO TO lms_select;
GRANT SELECT ON LMS_LEARNING_HISTORY TO lms_select;
GRANT SELECT ON LMS_ITEM_DETAIL TO lms_select;

3 rows selected.
```

3. Execute the statements which result from the query

```
SQL> GRANT SELECT ON LMS_STUDENT_INFO TO lms_select;

Grant succeeded.

SQL> GRANT SELECT ON LMS_LEARNING_HISTORY TO lms_select;

Grant succeeded.

SQL> GRANT SELECT ON LMS_ITEM_DETAIL TO lms_select;

Grant succeeded.
```

SCRIPTING WITH SQL PLUS

- Highlight results (DDL statement) and right click
- SQL*Plus will execute the statements

```
CMD
--------------------------------------------------------------------------------
GRANT SELECT ON LMS_STUDENT_INFO TO lms_select;
GRANT SELECT ON LMS_LEARNING_HISTORY TO lms_select;
GRANT SELECT ON LMS_ITEM_DETAIL TO lms_select;

3 rows selected.
```

```
SQL> GRANT SELECT ON LMS_STUDENT_INFO TO lms_select;

Grant succeeded.

SQL> GRANT SELECT ON LMS_LEARNING_HISTORY TO lms_select;

Grant succeeded.

SQL> GRANT SELECT ON LMS_ITEM_DETAIL TO lms_select;

Grant succeeded.

SQL>
```

Reval – recompiling invalid objects

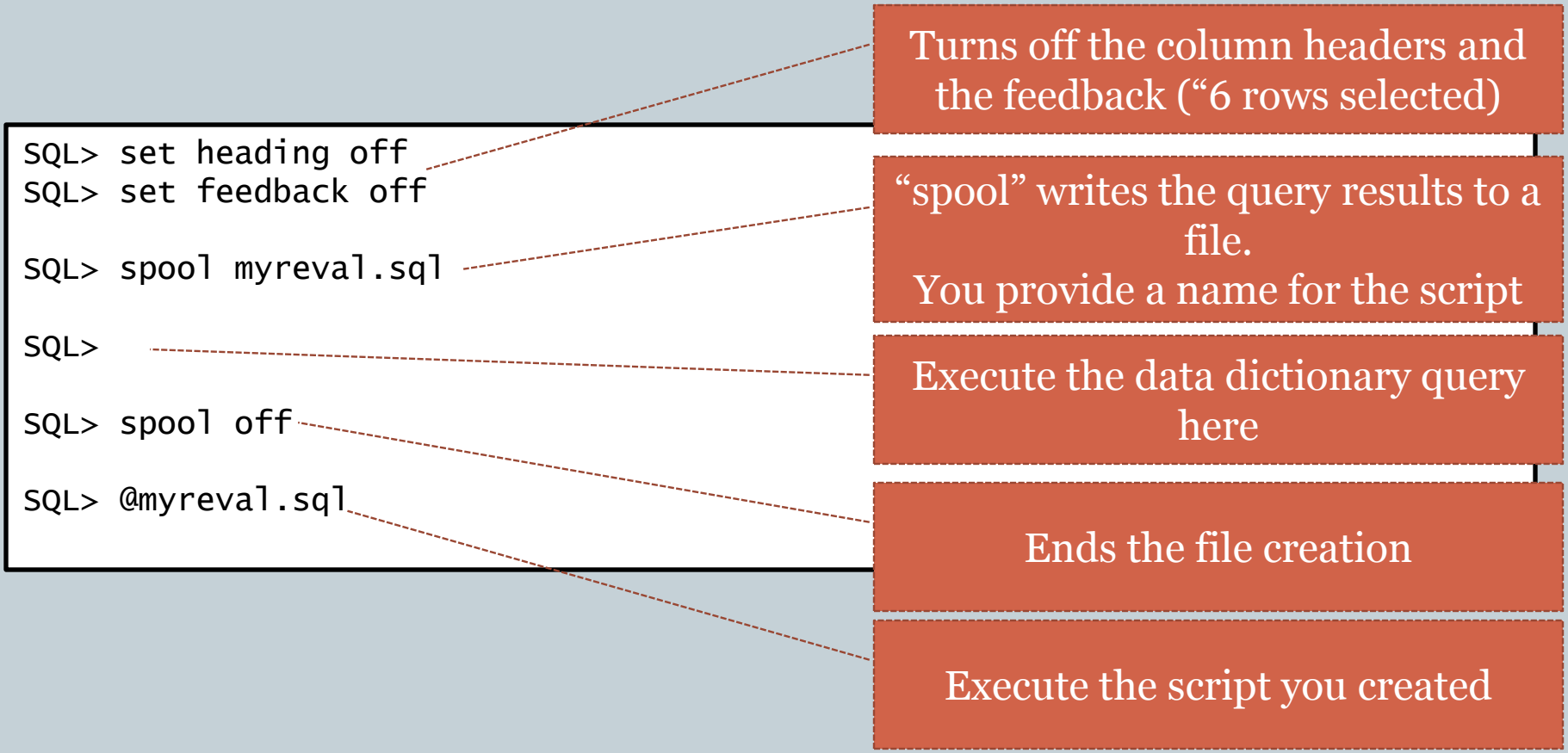
```
SELECT DISTINCT 'ALTER '
                || DECODE(object_type
                          , 'PACKAGE BODY', 'PACKAGE'
                          , object_type)
                || ' '
                || object_name
                || ' COMPILE'
                || DECODE(object_type,
                                          'PACKAGE BODY', ' BODY',
                                          'PACKAGE',      ' BODY')
                || ';'
FROM all_objects
WHERE object_type IN ('PACKAGE BODY', 'PACKAGE', 'PROCEDURE', 'FUNCTION', 'VIEW', 'TRIGGER')
  AND owner = 'HRP'
  AND object_name NOT LIKE 'BIN$%'
  AND status <> 'VALID';
```

```
'ALTER'||DECODE(OBJECT_TYPE,'PACKAGEBODY','PACKAGE',OBJECT_TYPE)||''||OBJECT_NAME||'COMPILE'||DECODE(O
--------------------------------------------------------------------------------------------------------
ALTER PROCEDURE ETL_NON_PROD_LABOR_EX_FULL COMPILE;
ALTER VIEW ELEMENT_FACT_COMP_PAID_CUBE COMPILE;
ALTER TRIGGER BALANCE_TRIG COMPILE;
ALTER FUNCTION UPDATE_INSERT_LOAD_ERRORS COMPILE;
ALTER PROCEDURE TMS_TEST_RESULTS_LOAD COMPILE;
ALTER PACKAGE SUMMARY_BG_TABLES_REFRESH_PKG COMPILE BODY;

6 rows selected.
```

heading

feedback

Building an executable script



Things to Remember

- The more you use it, the more use cases you'll discover!
- I built this presentation by pasting in queries that I wrote while doing my job
- Don't be afraid to query!

Agenda

- Why use the data dictionary?
- The data dictionary hierarchy
- Some useful data dictionary views
- Scripting / Building DDL

THANK-YOU!