

Chapter 14 Data Dictionary and Scripting

1

Tables in the Oracle Database

- User Tables
 - Collection of tables to store data
- Data Dictionary Tables
 - Collection of tables created and maintained by Oracle server
 - Not directly accessible
 - users typically access the data dictionary via **views**
 - Owned by the user SYS

2

Data Dictionary Views

- Dynamic Views
 - Begin with v\$
 - Are continuously updated to monitor system performance
 - Generally for DBA usage
 - Example

```
SELECT username FROM v$session;
```
- Static Views
 - Information including users, privileges granted to users, database object names, table constraints, and auditing information
 - 3 prefixes
 - **USER_** view information regarding user's own schema objects
 - **ALL_** view information on user's own objects and those GRANTED
 - **DBA_** view information regarding all objects (must be DBA)

3

Querying the Data Dictionary

- **USER_OBJECTS** (pg 620)
 - describes all objects owned by the current user
 - tables, views, sequences, indexes, synonyms, functions, procedures, triggers, packages, others

```
DESC USER_OBJECTS

SELECT object_name, object_type, created, status
FROM USER_OBJECTS
ORDER BY object_type, object_name;
```

- Note:
 - data dictionary views display object names in **UPPER** case

4

Who Has Created their Payment Table?

■ ALL_OBJECTS

- describes objects owned by the current user and objects the current user has privileges to use

```
DESC ALL_OBJECTS

SELECT owner, object_name,
       to_char(created, 'yyyy/mm/dd hh:mi')
FROM ALL_OBJECTS
WHERE object_name='PAYMENT'
      AND object_type='TABLE'
      AND created > (SYSDATE - 120)
ORDER BY 3;
```

5

Confirming Views

■ USER_VIEWS (pg 623, we used in Ch13)

- contains details of each view owned by the current user

```
DESC USER_VIEWS

SELECT view_name, text
FROM USER_VIEWS;
```

6

Confirming Tables

■ USER_TABLES (pg 621)

- describes base tables owned by the current user

```
DESC USER_TABLES
```

```
SELECT table_name, blocks, num_rows, avg_row_len,
       to_char(last_analyzed, 'DD-MON-RR HH24:MI')
FROM USER_TABLES
ORDER BY table_name;
```

Query Result x | All Rows Fetched: 33 in 0.234 seconds

TABLE_NAME	BLOCKS	NUM_ROWS	AVG_ROW_LEN	TO_CHAR(LAST_ANALYZED)
1 ACTOR	5	4	15	23-MAR-11 22:00
2 ARTICLE	5	33	51	18-APR-11 21:00
3 ARTICLE2	0	0	0	11-APR-11 21:00
4 COMIC	5	3	14	23-MAR-11 22:00
5 COURSE	5	30	65	24-JAN-11 22:00
6 CREDITCARD	5	6	29	27-APR-09 17:30

Gathering Statistics (lib, pg 804+)

- The Oracle Optimizer uses object statistics to determine how to efficiently execute SQL statements
- Oracle does not continuously update object statistics
- Oracle 10g schedules GATHER_STATS_JOB
 - collects statistics on database objects when the object has no previously gathered statistics or the existing statistics are stale because the underlying object has been modified significantly (eg: >10% of the rows)
- For how long are these statistics accurate?
- Can execute DBMS_STATS to gather statistics on the spot

```
EXEC DBMS_STATS.gather_table_stats('SCOTT', 'EMPLOYEES');
EXEC DBMS_STATS.gather_schema_stats(ownname => 'SCOTT');
EXEC DBMS_STATS.gather_table_stats('TTROLLEN', 'ARTICLE');
```

Query Result x | Statement Output x

```
anonymous block completed
anonymous block completed
anonymous block completed
```

8

Confirming Table Columns

- **USER_TAB_COLUMNS** (pg 622)
 - describes the columns of current user's tables and views

```
DESC USER_TAB_COLUMNS

SELECT column_name, data_type, data_precision,
       data_scale, avg_col_len
FROM USER_TAB_COLUMNS
WHERE table_name = 'ARTICLE';
```

	COLUMN_NAME	DATA_TYPE	DATA_PRECISION	DATA_SCALE	AVG_COL_LEN
1	ARTICLENUM	NUMBER		4	0
2	TITLE	VARCHAR2	(null)	(null)	28
3	TYPE	CHAR	(null)	(null)	4
4	ISSUE	DATE	(null)	(null)	8
5	LENGTH	NUMBER		5	0
6	WRITERID	CHAR	(null)	(null)	5

What Data Dictionary Views Are Available?

- **DICTIONARY** (pg 617)
 - shows the name & a comment for each data dictionary view the current user can use

```
DESC DICTIONARY
```

```
SELECT *
FROM DICTIONARY
WHERE UPPER(comments) LIKE '%TABLE%'
ORDER BY table_name;
```

Query Result x

SQL | Fetched 50 rows in 0.266 seconds

	TABLE_NAME	COMMENTS
1	ALL_ALL_TABLES	Description of all object and relational tables accessible to the user
2	ALL_APPLY_CONFLICT_COLUMNS	Details about conflict resolution on tables visible to the user
3	ALL_APPLY_DML_HANDLERS	Details about the dml handler on tables visible to the user
4	ALL_APPLY_KEY_COLUMNS	Alternative key columns for a STREAMS table visible to the user
5	ALL_APPLY_TABLE_COLUMNS	Details about the columns of destination table objects

What Do The Columns in a Data Dictionary View Mean?

- **DICTIONARY_COLUMNS** (pg 626)
 - shows the name & a comment for each column in a data dictionary view

```
DESC DICTIONARY_COLUMNS
```

```
SELECT column_name, comments
FROM DICTIONARY_COLUMNS
WHERE table_name = 'USER_TABLES'
AND column_name IN ('BLOCKS', 'AVG_ROW_LEN', 'NUM_ROWS', 'LAST_ANALYZED');
```

Statement Output x Query... x

SQL | All Rows Fetched: 4 in 0.015 seconds

	COLUMN_NAME	COMMENTS
1	NUM_ROWS	The number of rows in the table
2	BLOCKS	The number of used blocks in the table
3	AVG_ROW_LEN	The average row length, including row overhead
4	LAST_ANALYZED	The date of the most recent time this table was analyzed

Data Dictionary Practice

- Which data dictionary views seem to relate to constraints?

Confirming Constraints

- **USER_CONSTRAINTS** (pg 624+)
 - describes the constraints on tables owned by the current user
 - constraint type codes: C P R U V O

```
SELECT constraint_name, constraint_type,
       search_condition
FROM USER_CONSTRAINTS
WHERE table_name = 'WRITER';
```

Query Result x

SQL | All Rows Fetched: 6 in 0.031 seconds

CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION
1 WRITER_WRITER_ID_NN	C	"WRITERID" IS NOT NULL
2 WRITER_LN_NN	C	"LN" IS NOT NULL
3 WRITER_FN_NN	C	"FN" IS NOT NULL
4 WRITER_FREELANCER_CK	C	freelancer IN ('Y', 'N')
5 WRITER_WRITERID_PK	P	(null)
6 WRITER_CONTACT_FK	R	(null)

Viewing the Columns Associated with Constraints

- **USER_CONS_COLUMNS** (pg 626)
 - shows which column(s) a constraint operates on
 - especially useful for constraints with a system-assigned name

- **Practice Time**
 - what does **USER_CONS_COLUMNS** show for the ENROLLMENT table?

```
SELECT constraint_name, column_name
FROM USER_CONS_COLUMNS
WHERE table_name = 'WRITER';
```

Statement Output x Query Result x

SQL | All Rows Fetched: 6 in 0.016 seconds

CONSTRAINT_NAME	COLUMN_NAME
1 WRITER_CONTACT_FK	CONTACT
2 WRITER_WRITERID_PK	WRITERID
3 WRITER_FREELANCER_CK	FREELANCER
4 WRITER_FN_NN	FN
5 WRITER_LN_NN	LN
6 WRITER_WRITER_ID_NN	WRITERID

Joining Dictionary Views

- Related views can be joined via a common column
 - **CONSTRAINT_NAME** is common to **USER_CONSTRAINTS** and **USER_CONS_COLUMNS**
 - show details for WRITER table's constraints and columns they act upon

```
SELECT ucc.constraint_name, column_name,
       search_condition, constraint_type
FROM user_constraints uc, user_cons_columns ucc
WHERE ucc.constraint_name = uc.constraint_name
AND ucc.table_name = 'WRITER'
ORDER BY 1, 2;
```

Both	UCC	UC	UC
CONSTRAINT_NAME	COLUMN_NAME	SEARCH_CONDITION	CONSTRAINT_TYPE
1 WRITER_CONTACT_FK	CONTACT	(null)	R
2 WRITER_FN_NN	FN	"FN" IS NOT NULL	C
3 WRITER_FREELANCER_CK	FREELANCER	freelancer IN ('Y', 'N')	C
4 WRITER_LN_NN	LN	"LN" IS NOT NULL	C
5 WRITER_WRITERID_PK	WRITERID	(null)	P
6 WRITER_WRITER_ID_NN	WRITERID	"WRITERID" IS NOT NULL	C

Practice Time

- What would a join of **USER_CONSTRAINTS** and **USER_CONS_COLUMNS** show for the ENROLLMENT table?

ENROLLMENT		
STUDENT_ID (FK)(FK)	NUMBER(8,0)	NOT NULL
SECTION_ID (FK)(FK)	NUMBER(8,0)	NOT NULL
ENROLL_DATE	DATE	NOT NULL
FINAL_GRADE	NUMBER(3,0)	NULL
CREATED_BY	VARCHAR2(30)	NOT NULL
MODIFIED_BY	VARCHAR2(30)	NOT NULL
MODIFIED_DATE	DATE	NOT NULL

Object Dependencies

- Some objects are dependent on other objects
 - eg: a view depends on _____ and _____
- USER_DEPENDENCIES** (pg 624)
 - describes dependencies between views, procedures, packages, functions, and triggers owned by the current user

```
DESC USER_DEPENDENCIES
```

```
SELECT name, referenced_name,
       referenced_type, referenced_owner
FROM USER_DEPENDENCIES
WHERE name = 'ARTICLE_WRITER';
```

Statement Output x Query... x

SQL | All Rows Fetched: 2 in 0 seconds

	NAME	REFERENCED_NAME	REFERENCED_TYPE	REFERENCED_OWNER
1	ARTICLE_WRITER	ARTICLE	TABLE	TTROLLEN
2	ARTICLE_WRITER	WRITER	TABLE	TTROLLEN

17

Retrieving an Object's DDL (nib?)

- The Data Dictionary contains each object's definition
- Can use **DBMS_METADATA.GET_DDL** to extract the SQL statement that defines the object

```
SELECT DBMS_METADATA.GET_DDL('TABLE', 'RECEIPT', 'TTROLLEN')
FROM DUAL;
```



```
SELECT DBMS_METADATA.GET_DDL('SEQUENCE', 'EVENES_SEQ')
FROM DUAL;
```

```
SELECT DBMS_METADATA.GET_DDL('VIEW', 'ARTICLE_WRITER')
FROM DUAL;
```

18

Lab 14.2

Scripting and Reporting

- Changes to database objects will be developed and tested in a **development** testbed, then applied to the **production** schema
- What is a script?
 - a text file containing SQL statements and SQL*Plus commands
 - ensures the same series of DML/DDDL statements executed in production as testbed
- 2 Ways to run a script from SQL Developer
 - open the script file and run it  or [F5]
 - start command 

- The default script filename extension is .sql
- What is the default script location?
 - Preferences > Database > Worksheet Parameters > Select default path to look for scripts

19

Common SQL*Plus Commands (pg 860+)

- DESCRIBE**
- comments**
 - single-line `--` or `REM`
 - multi-line `/* */`
- SPOOL filename**
- SET ECHO {ON | OFF}**
 - controls whether or not the Script Output pane shows commands when a script executes
- SET DEFINE {& | c | ON | OFF}**
 - controls whether or not SQL Developer scans commands for substitution variables and replaces them with their values. ON changes the value of c back to the default &, not the most recently used c
 - sets the character used to prefix substitution variables to c
- SET VERIFY {ON | OFF}**
 - controls whether the the Script Output pane shows substitution variable details
- SET FEEDBACK {<n> | ON | OFF}**
 - displays the row count when a statement returns at least n rows
 - eg: '9 rows selected'
 - SET FEEDBACK OFF suppresses the row count and also suppresses statement confirmation messages
 - eg: 'Table created'
- ACCEPT**
- UNDEFINE**
- PAUSE**
- PROMPT**

20

SQL*Plus Substitution Variables

- What is a **variable**?
- In SQL*Plus and SQL Developer, **&** indicates that what follows is a **substitution variable**
- Substitution variables make SQL statements more **versatile**
- When a **substitution variable** is embedded in a SQL statement, SQL Developer substitutes the variable's **value** into the statement, prompting for its value if not previously created, before sending the statement to Oracle
- Can be created **explicitly**
 - ACCEPT
 - DEFINE
- Will be created **implicitly** if not previously created

21

Practice Time: Substitution Variables

- Open a new worksheet, then enter these statements

```
--constraints.sql
SET VERIFY ON

SELECT constraint_name, constraint_type, search_condition
FROM USER_CONSTRAINTS
WHERE table_name = '&v_table';
```

- Save this script as **constraints.sql**
- Run the script
- Note:
 - how substitution variables were resolved
 - row count is displayed

Enter Substitution Variable

V_TABLE:

OK Cancel

ACCEPT (pg 650+)

```
ACCEPT variable [NUMBER|CHAR|DATE] [FORMAT format]
[PROMPT text] [HIDE]
```

- A SQL*Plus command
 - regrettably, seems to have limited functionality in SQL Developer
- Defines a **variable**, its data type, and a prompt message

```
ACCEPT v_table CHAR PROMPT 'Which table?'
ACCEPT hired DATE FORMAT 'mm/dd/yyyy' -
DEFAULT '01/01/2010' PROMPT 'Enter hire date:'
ACCEPT v_length NUMBER PROMPT 'What size?'
ACCEPT pw CHAR PROMPT 'Password' HIDE
```

- The variable retains its value until exit SQL*Developer or the variable is explicitly **UNDEFINED**
 - can refer to the variable throughout the session **without getting prompted for each occurrence**

23

Demo: Substitution Variables

```
--constraints.sql
SET VERIFY OFF
SET FEEDBACK OFF
ACCEPT v_table CHAR PROMPT 'View constraint details for which table?'

Prompt Constraint Details for table &v_table:

SELECT constraint_name, constraint_type, search_condition
FROM USER_CONSTRAINTS
WHERE table_name = '&v_table';

SELECT COUNT(*) AS "&v_table constraint count is"
FROM USER_CONSTRAINTS
WHERE table_name = '&v_table' ;

SELECT column_name AS "&v_table's PRIMARY KEY is "
FROM user_constraints uc, user_cons_columns ucc
WHERE ucc.constraint_name = uc.constraint_name
AND ucc.table_name = UPPER('&v_table')
AND constraint_type = 'P';

UNDEFINE v_table
SET VERIFY ON
```

24

UNDEFINE

- A SQL*Plus command
- Deletes variables that were defined either explicitly (ACCEPT, DEFINE) or implicitly

```
UNDEFINE v_table  
  
UNDEFINE pw  
  
DEFINE
```

25

Generating Dynamic SQL

- What does all this concatenation produce?

```
SELECT 'ALTER TABLE ' || table_name ||  
      ' DISABLE CONSTRAINT ' || constraint_name || ','  
FROM USER_CONSTRAINTS  
WHERE table_name = 'ENROLLMENT';
```

```
1 'ALTERTABLE||TABLE_NAME||DISABLECONSTRAINT||CONSTRAINT_NAME|';  
1 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_MODIFIED_DATE_NULL;  
2 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_STUDENT_ID_NULL;  
3 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_ENROLL_DATE_NULL;  
4 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_MODIFIED_BY_NULL;  
5 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_SECTION_ID_NULL;  
6 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_CREATED_BY_NULL;  
7 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_CREATED_DATE_NULL;  
8 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_PK;  
9 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_STU_FK;  
10 ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_SECT_FK;
```

- Thus we can **use SQL to generate SQL**

26

Generate Dynamic SQL

- Place the generated rows to a text file to build a **script**

```
-- disable_enrollment.sql  
Prompt This script disables the constraints for table ENROLLMENT  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_MODIFIED_DATE_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_STUDENT_ID_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_ENROLL_DATE_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_MODIFIED_BY_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_SECTION_ID_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_CREATED_BY_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_CREATED_DATE_NULL;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_PK;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_STU_FK;  
ALTER TABLE ENROLLMENT DISABLE CONSTRAINT ENR_SECT_FK;
```

27