Introduction to Data Dictionary Views

Data Dictionary

Created by user

Created by oracle server (read only)

Oracle Server

Tables containing business data:

EMPLOYEES
DEPARTMENTS
LOCATIONS
JOB_HISTORY
. . .

Data dictionary views:

DICTIONARY
USER_OBJECTS
USER_TABLES
USER_TAB_COLUMNS
. . .
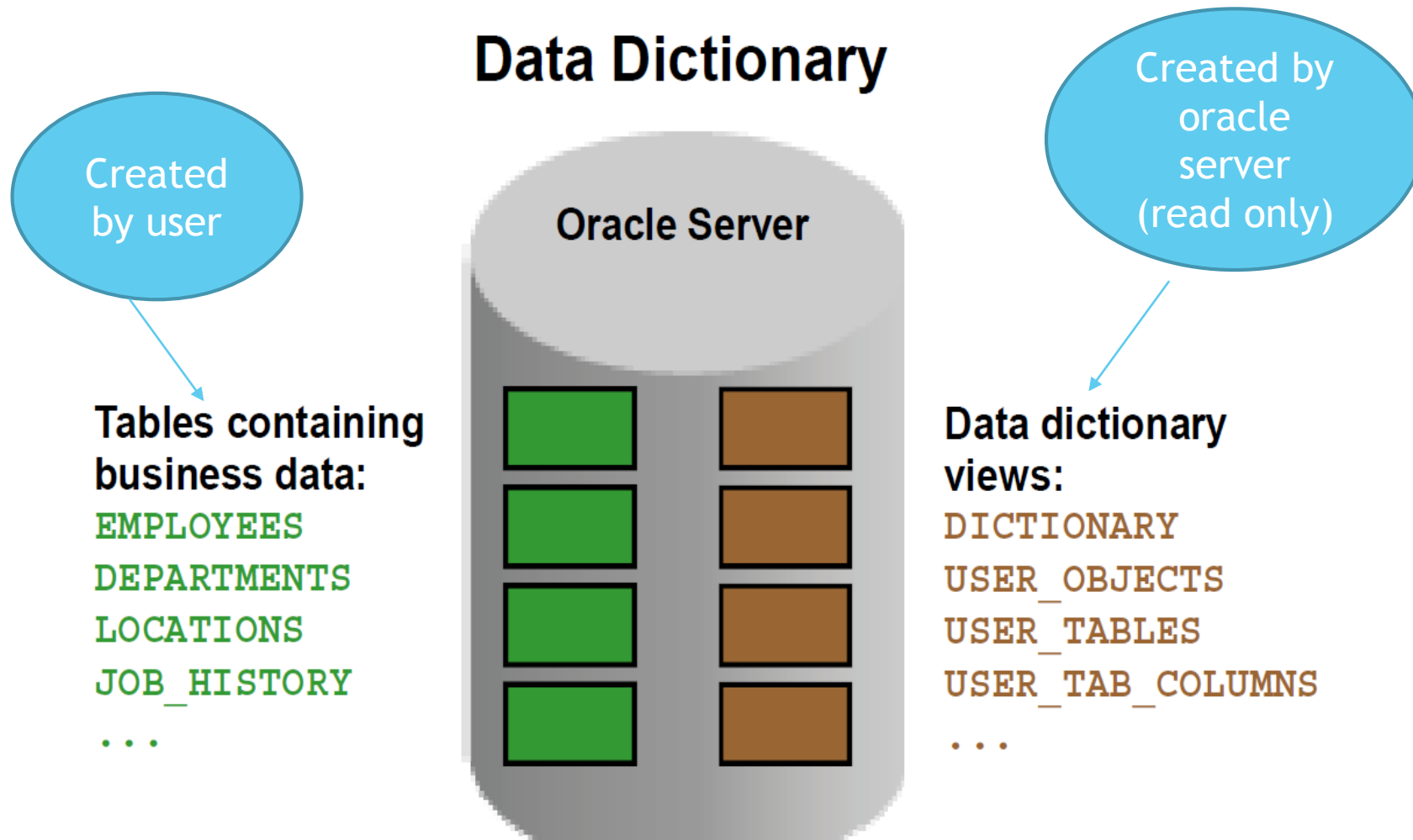
You use SQL statements to access the data dictionary. Because the data dictionary is read-only, you can issue only queries against its tables and views.

You can query the dictionary views that are based on the dictionary tables to find information such as:

- Definitions of all schema objects in the database (tables, views, indexes, synonyms, sequences, procedures, functions, packages, triggers, and so on)
- Default values for columns
- Integrity constraint information
- Names of Oracle users
- Privileges and roles that each user has been granted
- Other general database information

Data Dictionary Structure

Oracle Server

Consists of:
- Base tables
- User-accessible views
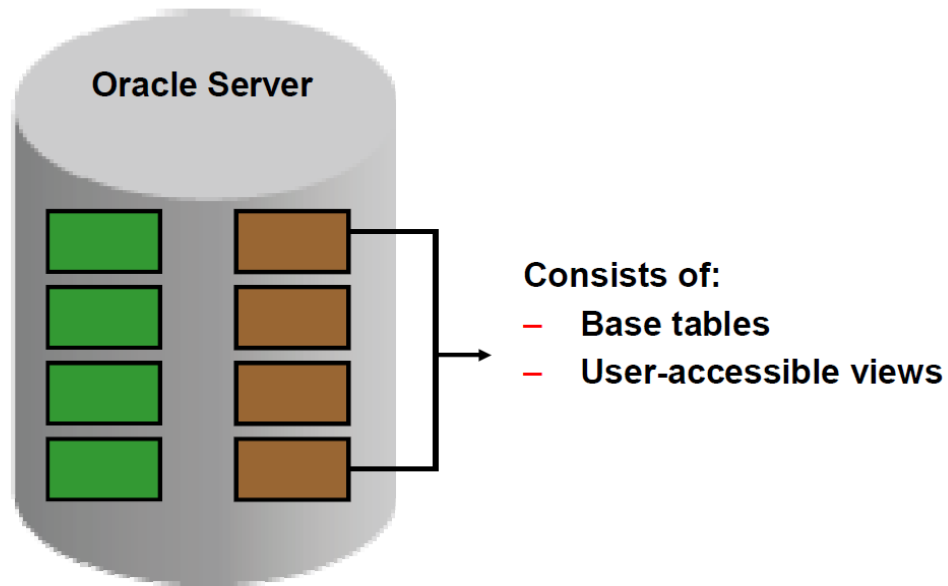
Underlying base tables store information about the associated database. Only the Oracle Server should write to and read from these tables. You rarely access them directly.

There are several views that summarize and display the information stored in the base tables of the data dictionary. These views decode the base table data into useful information (such as user or table names) using joins and WHERE clauses to simplify the information. Most users are given access to the views rather than the base tables.

The Oracle user SYS owns all base tables and user-accessible views of the data dictionary. No Oracle user should *ever* alter (UPDATE, DELETE, or INSERT) any rows or schema objects contained in the SYS schema, because such activity can compromise data integrity.
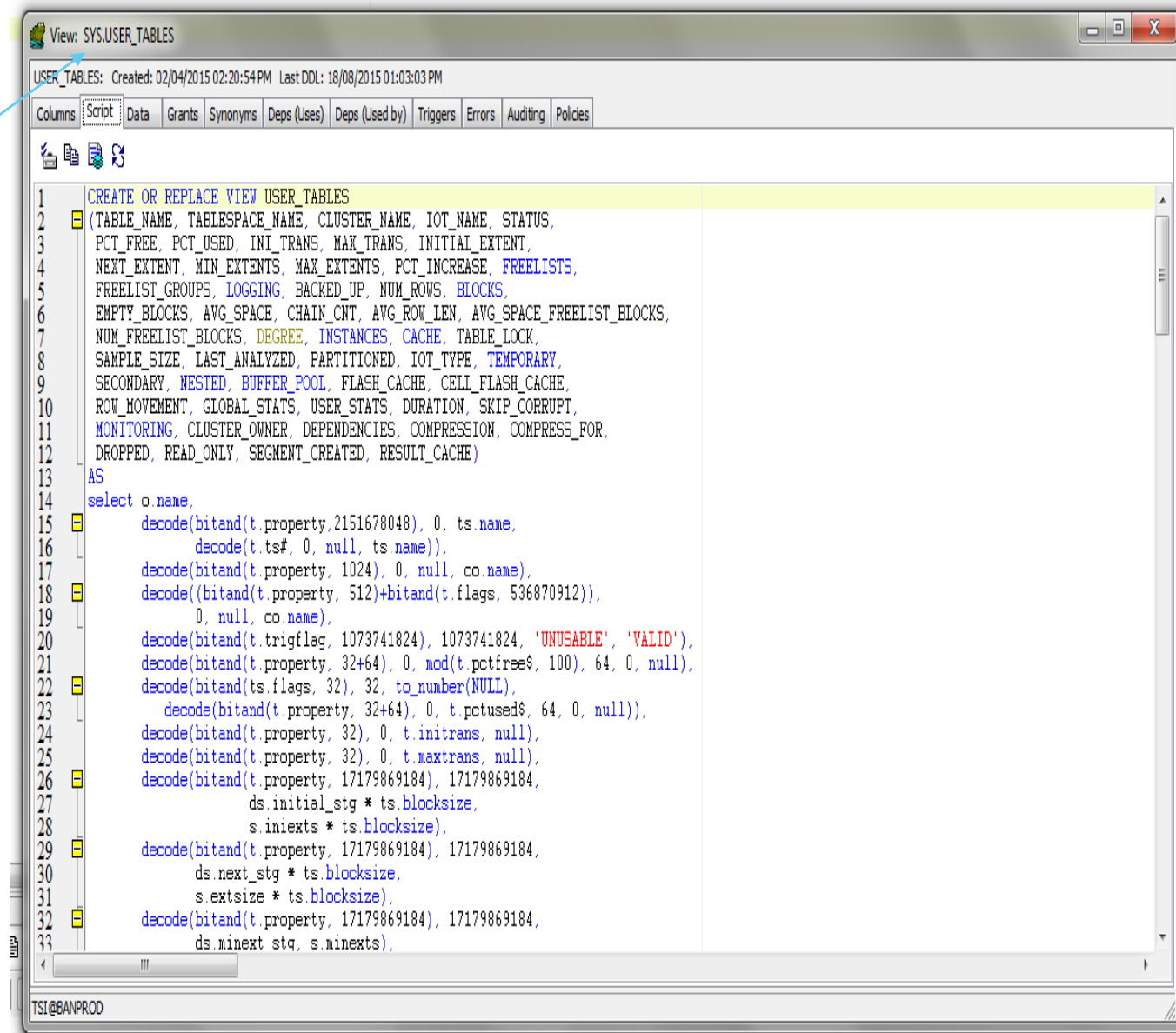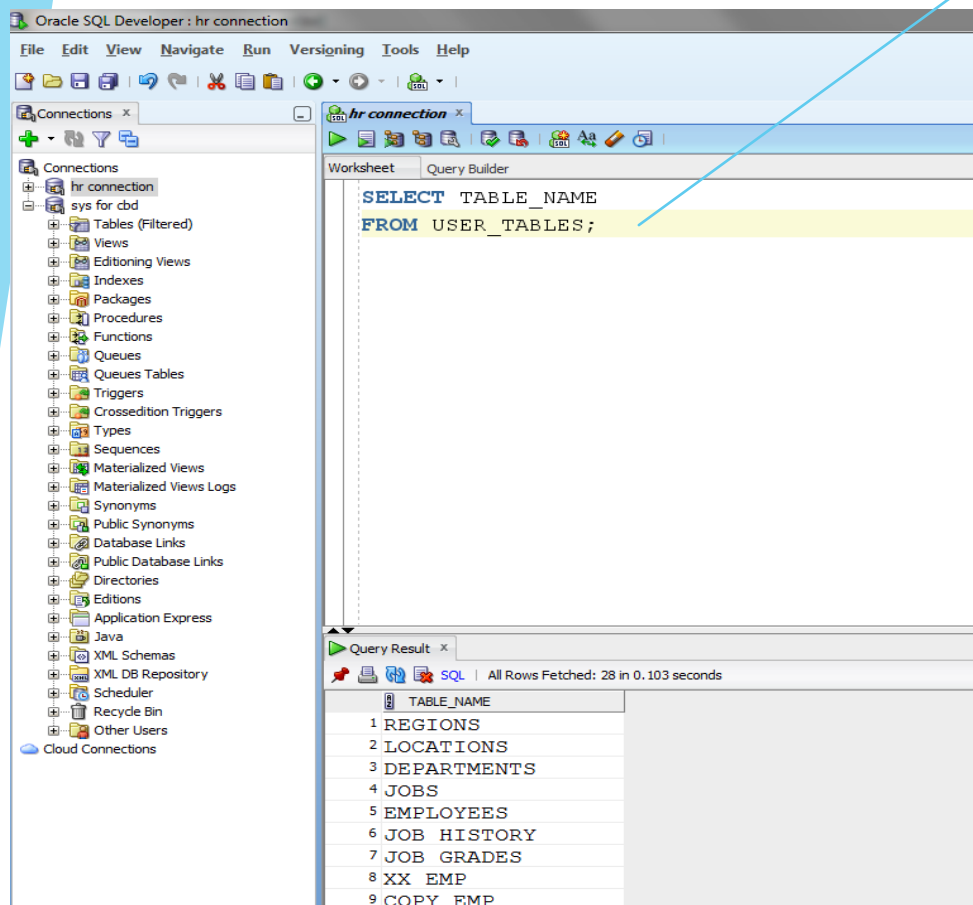
You always access the dictionary views and no need to know how it created by sys

View naming convention:

View Prefix	Purpose
USER	User's view (what is in your schema; what you own)
ALL	Expanded user's view (what you can access)
DBA	Database administrator's view (what is in everyone's schemas)
V$	Performance-related data

The data dictionary consists of sets of views. In many cases, a set consists of three views containing similar information and distinguished from each other by their prefixes. For example, there is a view named USER_OBJECTS, another named ALL_OBJECTS, and a third named DBA_OBJECTS.

These three views contain similar information about objects in the database, except that the scope is different. USER_OBJECTS contains information about objects that you own or you created. ALL_OBJECTS contains information about all objects to which you have access. DBA_OBJECTS contains information about all objects that are owned by all users. For views that are prefixed with ALL or DBA, there is usually an additional column in the view named OWNER to identify who owns the object.

There is also a set of views that is prefixed with V$. These views are dynamic in nature and hold information about performance. Dynamic performance tables are not true tables, and they should not be accessed by most users. However, database administrators can query and create views on the tables and grant access to those views to other users. This course does not go into details about these views.

Start with DICTIONARY. It contains the names and descriptions of the dictionary tables and views.

```
DESCRIBE DICTIONARY
```

```
DESCRIBE dictionary
Name          Null Type
----------- ---- ---------------
TABLE_NAME         VARCHAR2(128)
COMMENTS           VARCHAR2(4000)
```

```
SELECT *
FROM    dictionary
WHERE   table_name = 'USER_OBJECTS';
```

TABLE_NAME	COMMENTS
1 USER_OBJECTS	Objects owned by the user

Note: The names in the data dictionary are in uppercase.

USER_OBJECTS and ALL_OBJECTS Views

USER_OBJECTS:
- Query USER_OBJECTS to see all the objects that you own.
- Using USER_OBJECTS, you can obtain a listing of all object names and types in your schema, plus the following information:
  - Date created
  - Date of last modification
  - Status (valid or invalid)

ALL_OBJECTS:
- Query ALL_OBJECTS to see all the objects to which you have access.

You can query the USER_OBJECTS view to see the names and types of all the objects in your schema. There are several columns in this view:

- **OBJECT_NAME:** Name of the object
- **OBJECT_ID:** Dictionary object number of the object
- **OBJECT_TYPE:** Type of object (such as TABLE, VIEW, INDEX, SEQUENCE)
- **CREATED:** Time stamp for the creation of the object
- **LAST_DDL_TIME:** Time stamp for the last modification of the object resulting from a data definition language (DDL) command
- **STATUS:** Status of the object (VALID, INVALID, or N/A)
- **GENERATED:** Was the name of this object system-generated? (Y|N)

USER_OBJECTS View

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

The OBJECT_TYPE column holds the values of either TABLE, VIEW, SEQUENCE, INDEX, PROCEDURE, FUNCTION, PACKAGE, or TRIGGER.

The STATUS column holds a value of VALID, INVALID, or N/A. Although tables are always valid, the views, procedures, functions, packages, and triggers may be invalid.

The CAT View

For a simplified query and output, you can query the CAT view. This view contains only two columns: TABLE_NAME and TABLE_TYPE. It provides the names of all your INDEX, TABLE, CLUSTER, VIEW, SYNONYM, SEQUENCE, or UNDEFINED objects.

Note: CAT is a synonym for USER_CATALOG—a view that lists tables, views, synonyms and sequences owned by the user.

Table Information

USER_TABLES:

```
DESCRIBE user_tables
```

```
DESCRIBE user_tables
Name                          Null      Type
----------------------------- --------- -------------
TABLE_NAME                    NOT NULL  VARCHAR2(128)
TABLESPACE_NAME                         VARCHAR2(30)
CLUSTER_NAME                            VARCHAR2(128)
IOT_NAME                                VARCHAR2(128)
```

...

```
SELECT table_name
FROM   user_tables;
```

	TABLE_NAME
1	REGIONS
2	LOCATIONS
3	DEPARTMENTS
4	JOBS
5	EMPLOYEES
6	JOB_HISTORY

You can use the USER_TABLES view to obtain the names of all your tables. The USER_TABLES view contains information about your tables. In addition to providing the table name, it contains detailed information about the storage.

The TABS view is a synonym of the USER_TABLES view. You can query it to see a listing of tables that you own:

```
SELECT table_name
FROM   tabs;
```

Column Information

USER_TAB_COLUMNS:

```
DESCRIBE user_tab_columns
```

```
Name                Null       Type
------------------  --------   ---------------
TABLE_NAME          NOT NULL   VARCHAR2(128)
COLUMN_NAME         NOT NULL   VARCHAR2(128)
DATA_TYPE                      VARCHAR2(128)
DATA_TYPE_MOD                  VARCHAR2(3)
DATA_TYPE_OWNER               VARCHAR2(128)
DATA_LENGTH         NOT NULL   NUMBER
DATA_PRECISION                NUMBER
DATA_SCALE                    NUMBER
NULLABLE                      VARCHAR2(1)
```

You can query the USER_TAB_COLUMNS view to find detailed information about the columns in your tables. Although the USER_TABLES view provides information about your table names and storage, detailed column information is found in the USER_TAB_COLUMNS view.

This view contains information such as:

- Column names
- Column data types
- Length of data types
- Precision and scale for NUMBER columns
- Whether nulls are allowed (Is there a NOT NULL constraint on the column?)
- Default value

Constraint Information

- USER_CONSTRAINTS describes the constraint definitions on your tables.
- USER_CONS_COLUMNS describes columns that are owned by you and that are specified in constraints.

```
DESCRIBE user_constraints
```

```
DESCRIBE user_constraints
Name                     Null       Type
------------------       --------   -------------
OWNER                               VARCHAR2(128)
CONSTRAINT_NAME          NOT NULL   VARCHAR2(128)
CONSTRAINT_TYPE                     VARCHAR2(1)
TABLE_NAME               NOT NULL   VARCHAR2(128)
SEARCH_CONDITION                    LONG()
R_OWNER                             VARCHAR2(128)
R_CONSTRAINT_NAME                   VARCHAR2(128)
DELETE_RULE                         VARCHAR2(9)
STATUS                              VARCHAR2(8)
```

USER_CONSTRAINTS: Example

```
SELECT  constraint_name, constraint_type,
        search_condition, r_constraint_name,
        delete_rule, status
FROM    user_constraints
WHERE   table_name = 'EMPLOYEES';
```

	CONSTRAINT_NAME	CONSTRAINT_TYPE	SEARCH_CONDITION	R_CONSTRAINT_NAME	DELETE_RULE	STATUS
1	EMP_MANAGER_FK	R	(null)	EMP_EMP_ID_PK	NO ACTION	ENABLED
2	EMP_JOB_FK	R	(null)	JOB_ID_PK	NO ACTION	ENABLED
3	EMP_DEPT_FK	R	(null)	DEPT_ID_PK	NO ACTION	ENABLED
4	EMP_EMP_ID_PK	P	(null)	(null)	(null)	ENABLED
5	EMP_EMAIL_UK	U	(null)	(null)	(null)	ENABLED
6	EMP_SALARY_MIN	C	salary > 0	(null)	(null)	ENABLED
7	EMP_JOB_NN	C	"JOB_ID" IS NOT NULL	(null)	(null)	ENABLED
8	EMP_HIRE_DATE_NN	C	"HIRE_DATE" IS NOT NULL	(null)	(null)	ENABLED
9	EMP_EMAIL_NN	C	"EMAIL" IS NOT NULL	(null)	(null)	ENABLED
10	EMP_LAST_NAME_NN	C	"LAST_NAME" IS NOT NULL	(null)	(null)	ENABLED

The CONSTRAINT_TYPE can be:

- C (check constraint on a table, or NOT NULL)
- P (primary key)
- U (unique key)
- R (referential integrity)
- V (with check option, on a view)
- O (with read-only, on a view)

The DELETE_RULE can be:

- **CASCADE:** If the parent record is deleted, the child records are deleted, too.
- **SET NULL:** If the parent record is deleted, change the respective child record to null.
- **NO ACTION:** A parent record can be deleted only if no child records exist.

The STATUS can be:

- **ENABLED:** Constraint is active.
- **DISABLED:** Constraint is made not active.

```
DESCRIBE user_cons_columns
```

```
DESCRIBE user_cons_columns
Name              Null     Type
---------------   -------- --------------
OWNER             NOT NULL VARCHAR2(128)
CONSTRAINT_NAME   NOT NULL VARCHAR2(128)
TABLE_NAME        NOT NULL VARCHAR2(128)
COLUMN_NAME                VARCHAR2(4000)
POSITION                   NUMBER
```

```
SELECT  constraint_name, column_name
FROM    user_cons_columns
WHERE   table_name = 'EMPLOYEES';
```

	CONSTRAINT_NAME	COLUMN_NAME
1	EMP_LAST_NAME_NN	LAST_NAME
2	EMP_EMAIL_NN	EMAIL
3	EMP_HIRE_DATE_NN	HIRE_DATE
4	EMP_JOB_NN	JOB_ID
5	EMP_SALARY_MIN	SALARY
6	EMP_EMAIL_UK	EMAIL
7	EMP_EMP_ID_PK	EMPLOYEE_ID
8	EMP_DEPT_FK	DEPARTMENT_ID
9	EMP_JOB_FK	JOB_ID
10	EMP_MANAGER_FK	MANAGER_ID

Note: A constraint may apply to more than one column.

You can also write a join between USER_CONSTRAINTS and USER_CONS_COLUMNS to create customized output from both tables.

Adding Comments to a Table

- You can add comments to a table or column by using the COMMENT statement:

```
COMMENT ON TABLE employees
IS 'Employee Information';
```

```
COMMENT ON COLUMN employees.first_name
IS 'First name of the employee';
```

- Comments can be viewed through the data dictionary views:
  - ALL_COL_COMMENTS
  - USER_COL_COMMENTS
  - ALL_TAB_COMMENTS
  - USER_TAB_COMMENTS

You can add a comment of up to 4,000 bytes about a column, table, view

▶Thank You