1

Theory

Theory

Database Objects

Lets discuss few of the data structures available in oracle:

- Table: Stores data
- View: Subset of data from one or more tables
- Sequence: Generates primary key values
- Index: Improves the performance of some queries
- Synonym: Gives alternative names to objects

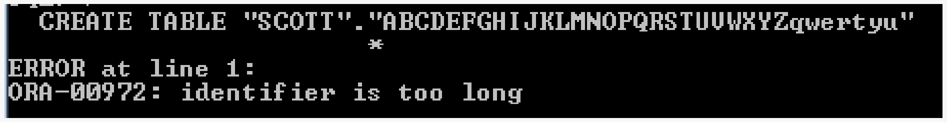
Oracle Table Structure

We do not specify the table size on creation so ultimately it will be depending on the total space available for the whole database.

SQL Developer

Naming Conventions

1. Table names and column names must begin with a letter and can be 1-30 characters Long other wise “IDENTIGIER IS TOO LONG” message will generate by SQL.



```
 CREATE TABLE "SCOTT"."ABCDEFGHIJKLMNOPQRSTUVWXYZqwertyu"
                       *
ERROR at line 1:
ORA-00972: identifier is too long
```

2. Names must contain only the characters A-Z, a-z, 0-9, _(underscore), $, and # (legal characters, but their use is discouraged).

3. Names must not duplicate the name of another object owned by the same **Oracle Server** & Names must not be a oracle server reserved word

2

Creating & Altering Tables

Creating & Altering Table

Create Table Statement

User must have create table privileges and storage space available before creating tables.

Syntax:

CREATE TABLE [schema .] table name
(***column_name*** datatype [DEFAULT expr] [, …]);

What is a schema?

*In Oracle, users and **schemas** are essentially the same thing. You can consider that a user is the account you use to connect to a database, and a schema is the set of objects (tables, views, synonyms ,sequences stored procedures ,indexes , clusters and database links) that belong to that account.*

What is a Database link?

A database link is a schema's object in one database that enables you to access objects of another database

Creating & Altering Table

Create Table Statement (cont)

Syntax:

CREATE TABLE [schema .] table name

(***column_name*** datatype [DEFAULT expr] [, …]);

What is a ***DEFAULT expr*** or ***Default option***?

A column can be given a default value by using default expression or default option

In last lab we have seen if we will not insert a value in a column then Null value by default will insert in that column. But after defining Default expression or option that default value will insert rather than NULL value.

Default value can be a literal{'123' or 'abc'} or can be a expression(case statement) or can be a function or a pseudo column like SYSDATE or USER(select User from dual return User name).Default expression must also match datatype of column.

Default value can not be name of another column so the pseudo columns such as NEXTVAL(sequencename.nextval tell next value of sequence) or CURRVAL(sequencename.nextval tells current value of a sequence) cant be Default expression.

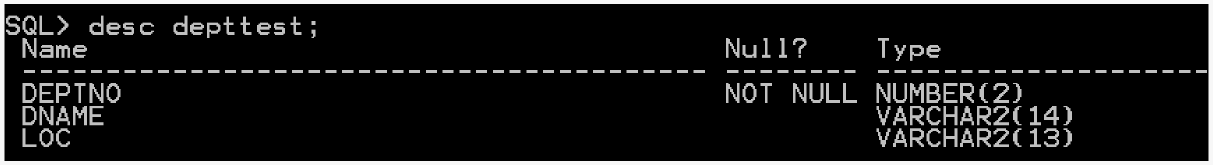
Creating & Altering Table

Create Table Example (video)

Creating & Altering Table

Creating a table is a DDL statement and as we discussed in last lab that an automatic commit take place when statement is executed.

In order to confirm the creation of the table you may write **DESC table** command



```
SQL> desc depttest;
 Name                                      Null?    Type
 ----------------------------------------- -------- --------------------
 DEPTNO                                    NOT NULL NUMBER(2)
 DNAME                                              VARCHAR2(14)
 LOC                                                VARCHAR2(13)
```

3

SQL Datatypes

SQL DataTypes

DATATYPE	DESCRIPTION
VARCHAR2(size)	Variable-length character data means storage size of the char value is the actual length of the data entered (A maximum size must be specified. Default and minimum size is 1; maximum size is 4000)
CHAR(size)	Fixed-length character data means the storage size of the char value is equal to the maximum size for this column (Default and minimum size is 1; maximum size is 2000)
NUMBER(p, s)	Number having precision p and scale s (The precision is the total number of decimal digits before the decimal point and the scale is the number of digits after the decimal point. The precision can range from 1 to 38 and the scale can range from -84 to 127.) Positive scale identifies the number of digits to the right of the decimal point; negative scale identifies the number of digits to the left of the decimal point that can be rounded up or down.if no scale is defined then scale is zero. See examples below.
DATE	Date and time values between January 1, 4712 B.C. (Before Christ)and December 31, 9999 A.D.
RAW(size)	stores ***binary*** data of length size (A maximum size must be specified. Maximum size is 2000 bytes.)
LONG RAW	Raw binary data of variable length up to 2 gigabytes
LONG	Variable length character data up to 2 gigabytes
CLOB	Single-byte character data up to 4 gigabytes
BLOB	Binary data up to 4 gigabytes
BFILE	Binary data stored in an external file; up to 4 gigabytes

SQL DataTypes

Number examples to understand Number(precision , Scale)

Here -2 says I need first right two digits to replace with “00” and 2 which occurs in range 1 will not impact to 1 So 1 will remain as it is.

INPUT	How We Defined Datatype	Stored As
7,456,123.89	NUMBER	7456123.89
7,456,123.89	NUMBER(*,1)	7456123.9 // will round scale to defined numbers .In this case, the precision is 38(maximum)
7,456,123.89	NUMBER(9)	7456124
7,456,123.89	NUMBER(9,2)	7456123.89
7,456,123.89	NUMBER(9,1)	7456123.9
7,456,123.89	NUMBER(6)	(not accepted, exceeds precision)
7,456,123.89	NUMBER(7,-2)	7456100
7,456,123.89	NUMBER	7456123.89

Long Data type

Do not create tables with LONG columns. Use LOB columns (CLOB, NCLOB, BLOB) instead. LONG columns are supported only for backward compatibility. Oracle also recommends that you convert existing LONG columns to LOB columns. LOB columns are subject to far fewer restrictions than LONG columns. Large Characters are directly stored in column of a database

SQL DataTypes

LOB Data types

CLOB belongs to the LOB Family. LOB is known as **Large Objects.** Used to store data in large size. It includes following data types:

- BFILE
- BLOB
- CLOB
- NCLOB

Basically they are used to store the blocks of unstructured data. Like heavy text, graphic images, videos and sounds. Can store only **maximum** of Four gigabytes

BFILE: Store large binary objects as file in operating system. These files are stored outside the database. Every BFILE stores a file locator which points to a large binary file on System. They cant be use in transactions means are only read-only.

BLOB: Every BLOB stores a locator in a database which points to a large binary objects. They are used in transactions means are recoverable and can be replicated.

SQL DataTypes

LOB Data types(cont.)

CLOB: Store **large blocks of character data** in database. It also stores a locator which points to A large block of character data. maximum size is of 4 gigabytes. They are also used in transactions means are recoverable and can be replicated.

NCLOB: same as CLOB but stores large block of Unicode characters data. Difference between **nchar (stores Unicode characters) & char** is same as difference between **CLOB & NLOB**.

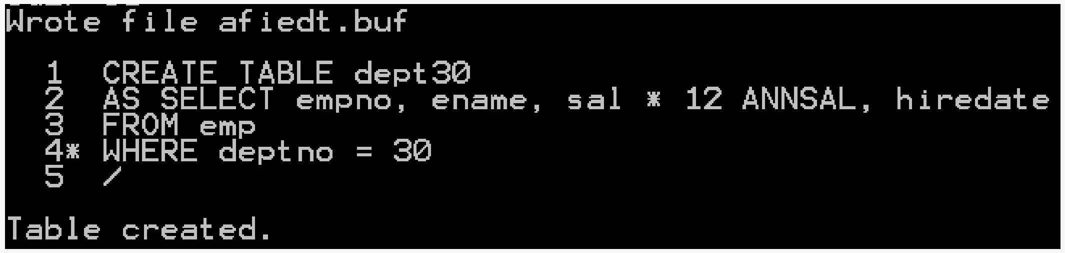
4

Creating Table Using Subquery

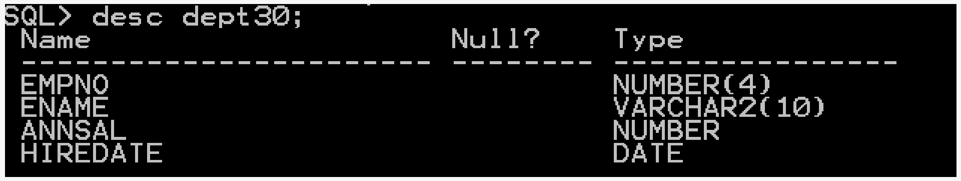
Creating tables using Subquery

Example : *create a new table named as* ***DEPT30*** *that have same structure of employee table and have data of all department 30's employees with annual salary as their basic salary.*

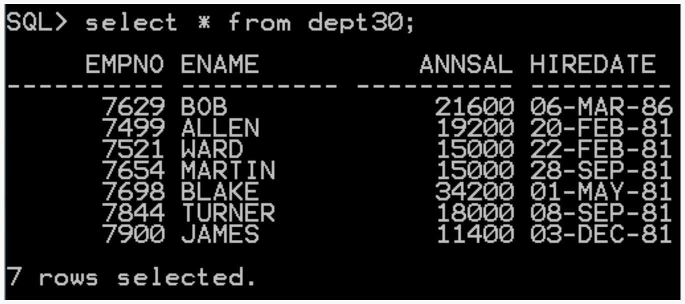
```
CREATE TABLE dept30
AS SELECT empno, ename, sal * 12 ANNSAL, hiredate
FROM emp
WHERE deptno = 30;
```



```
Wrote file afiedt.buf

  1  CREATE TABLE dept30
  2  AS SELECT empno, ename, sal * 12 ANNSAL, hiredate
  3  FROM emp
  4* WHERE deptno = 30
  5  /

Table created.
```



```
SQL> desc dept30;
 Name                          Null?    Type
 ----------------------------- -------- ------------------
 EMPNO                                  NUMBER(4)
 ENAME                                  VARCHAR2(10)
 ANNSAL                                 NUMBER
 HIREDATE                               DATE
```



```
SQL> select * from dept30;

     EMPNO ENAME          ANNSAL HIREDATE
---------- ---------- ---------- ---------
      7629 BOB             21600 06-MAR-86
      7499 ALLEN           19200 20-FEB-81
      7521 WARD            15000 22-FEB-81
      7654 MARTIN          15000 28-SEP-81
      7698 BLAKE           34200 01-MAY-81
      7844 TURNER          18000 08-SEP-81
      7900 JAMES           11400 03-DEC-81

7 rows selected.
```

5

Alter table Structure

Alter Table Structure

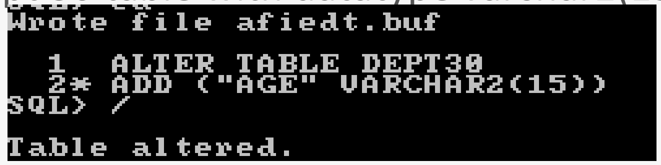
Alter table statement is used for:

<u>Add a new column in a table</u>

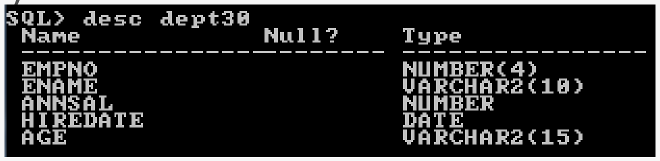
Add a new ***<u>AGE</u>*** column in dept 30 table with datatype varchar2(15).

ALTER TABLE DEPT30

ADD ("AGE" VARCHAR2(15))



```
Wrote file afiedt.buf

  1  ALTER TABLE DEPT30
  2* ADD ("AGE" VARCHAR2(15))
SQL> /

Table altered.
```



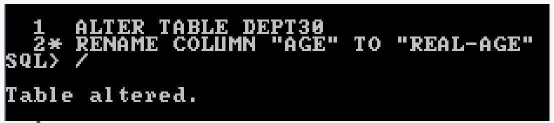
```
SQL> desc dept30
 Name                          Null?    Type
 ----------------------------- -------- ---------------
 EMPNO                                  NUMBER(4)
 ENAME                                  VARCHAR2(10)
 ANNSAL                                 NUMBER
 HIREDATE                               DATE
 AGE                                    VARCHAR2(15)
```

<u>Modify an existing column</u>

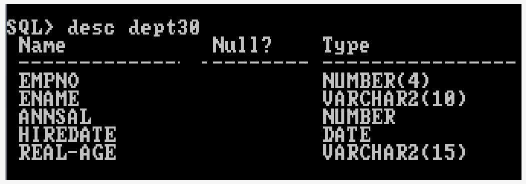
Change name of ***<u>AGE</u>*** column to ***<u>REAL-AGE</u>*** in dept 30 table with datatype

ALTER TABLE DEPT30

RENAME COLUMN “AGE" TO “REAL-AGE" ;



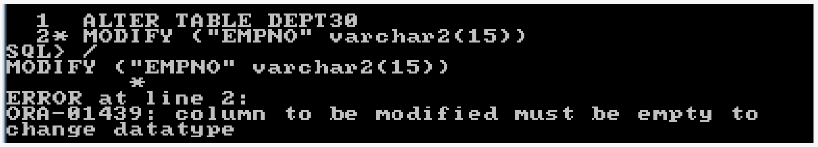
```
  1  ALTER TABLE DEPT30
  2* RENAME COLUMN "AGE" TO "REAL-AGE"
SQL> /

Table altered.
```



```
SQL> desc dept30
 Name                Null?    Type
 ------------------- -------- ----------------
 EMPNO                        NUMBER(4)
 ENAME                        VARCHAR2(10)
 ANNSAL                       NUMBER
 HIREDATE                     DATE
 REAL-AGE                     VARCHAR2(15)
```

Change the datatype of AGE column to number

Note: *<u>Before modifying the column datatype you must be sure that column is empty</u>*

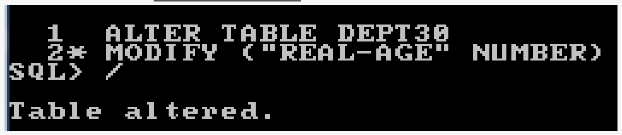


```
  1  ALTER TABLE DEPT30
  2* MODIFY ("EMPNO" varchar2(15))
SQL> /
MODIFY ("EMPNO" varchar2(15))
         *
ERROR at line 2:
ORA-01439: column to be modified must be empty to
change datatype
```

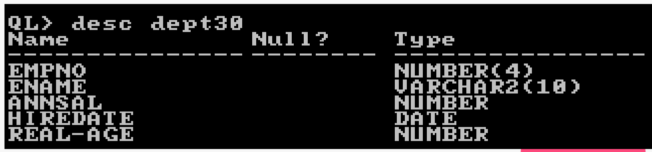
Change datatype of ***<u>REAL-AGE</u>*** column to ***<u>Number</u>*** in dept 30 table

ALTER TABLE DEPT30

MODIFY (“REAL-AGE " NUMBER);



```
  1  ALTER TABLE DEPT30
  2* MODIFY ("REAL-AGE" NUMBER)
SQL> /

Table altered.
```

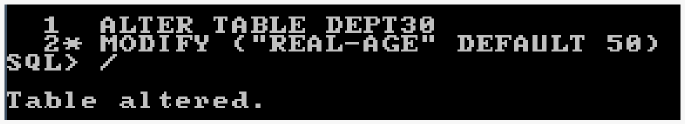


```
QL> desc dept30
Name                          Null?    Type
----------------------------- -------- ----------------
EMPNO                                  NUMBER(4)
ENAME                                  VARCHAR2(10)
ANNSAL                                 NUMBER
HIREDATE                               DATE
REAL-AGE                               NUMBER
```

Alter Table Structure

Define a default value for new column

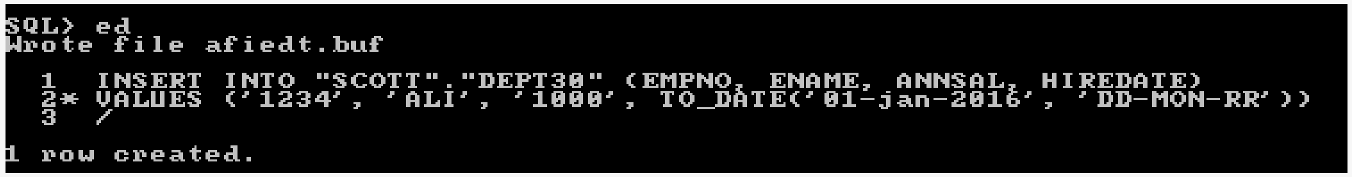
ALTER TABLE DEPT30
MODIFY ("REAL-AGE" DEFAULT 50);



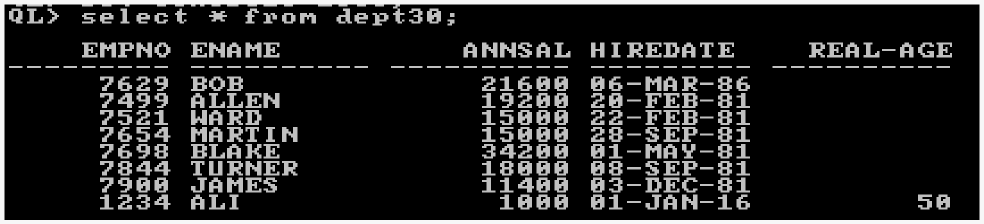
```
  1  ALTER TABLE DEPT30
  2* MODIFY ("REAL-AGE" DEFAULT 50)
SQL> /

Table altered.
```

Then see what happens when I insert record as :

INSERT INTO "SCOTT"."DEPT30" (EMPNO, ENAME, ANNSAL, HIREDATE) VALUES ('1234', 'ALI', '1000', TO_DATE('01-jan-2016', 'DD-MON-RR'))



```
SQL> ed
Wrote file afiedt.buf

  1  INSERT INTO "SCOTT"."DEPT30" (EMPNO, ENAME, ANNSAL, HIREDATE)
  2* VALUES ('1234', 'ALI', '1000', TO_DATE('01-jan-2016', 'DD-MON-RR'))
  3  /

1 row created.
```



```
QL> select * from dept30;

     EMPNO ENAME          ANNSAL HIREDATE    REAL-AGE
---------- ---------- ---------- --------- ----------
      7629 BOB             21600 06-MAR-86
      7499 ALLEN           19200 20-FEB-81
      7521 WARD            15000 22-FEB-81
      7654 MARTIN          15000 28-SEP-81
      7698 BLAKE           34200 01-MAY-81
      7844 TURNER          18000 08-SEP-81
      7900 JAMES           11400 03-DEC-81
      1234 ALI              1000 01-JAN-16         50
```

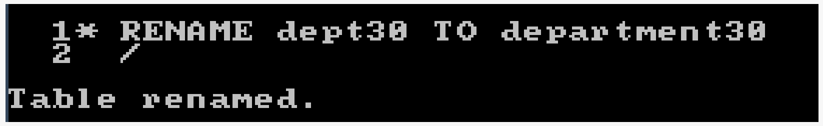
6

Changing the name of an object

Changing the name of an object

To change the name of a table, view, sequence or synonym we use RENAME statement as:

RENAME dept30 TO department30 ;



```
  1* RENAME dept30 TO department30
  2  /

Table renamed.
```

Drop table

Drop Table

The DROP TABLE statement removes the definition of an Oracle table. The DROP TABLE statement, once executed, is irreversible means commit as soon as it executes.
To drop the table DEPT30,

```
DROP TABLE DEPT30;
```

8

Constraints

Constraints

Oracle uses constraints to prevent invalid data entry into the tables. Constraints are used for following purposes:

➢Enforce rules at the table level whenever a row is inserted, updated, or deleted from that table. The constraint must be satisfied for the operation to succeed.

➢Prevent the deletion of a table if there are dependencies from other tables.

➢ Provide rules for Oracle tools, such as Oracle Developer. Means As a database developer, you need to know how to implement the data model that the application requires, how to implement the rules for data integrity, and how to implement the specified functions for accessing and manipulating the application data.

Constraints

Let see few constraints type in oracle:

DATATYPE	DESCRIPTION
NOT NULL	Specifies that this column may not contain a null value
UNIQUE	Specifies a column or combination of columns whose values must be unique for all rows in the table
PRIMARY KEY	Uniquely identifies each row of the table
FOREIGN KEY	Establishes and enforces a foreign key relationship between the column and a column of the referenced table
CHECK	Specifies a condition that must be true

Constraints Guidelines

➢All constraints are stored in the data dictionary. Data dictionary is a meta data. Means all tables and constraints meta data is present somewhere in sysdba table.

Note : Conn /as sysdba (password is also sysdba) (Video for datadictionary)

➢Specify a name for the constraint If you omit this identifier then Oracle Database generates a name with the form SYS_Cn where n is an integer so that the constraint name is unique

Constraints

Constraints Guidelines(cont)

Better to use following conventions when naming constraints:

Constraint type	Abbreviation
References (foreign key)	fk
unique	un
primary key	pk
check	ck
not null	nn
index	idx

➢Better to create constraints at the time of table creation or just after the table creation

Our scott tables constraints

```
CREATE TABLE DEPT (
DEPTNO          NUMBER(2) constraint DEPT_DEPTNO_PK PRIMARY KEY,
DNAME           VARCHAR2(14),
LOC             VARCHAR2(13),
CONSTRAINT      DEPT_DNAME_UK                UNIQUE(DNAME)      );
```

UK represents that it is **unique constraint** means all department must have unique name

Constraints

Our scott tables constraints(cont.)

```
CREATE TABLE EMP (
EMPNO           NUMBER(4) CONSTRAINT EMP_EMPNO_PK PRIMARY KEY,
ENAME           VARCHAR2(10) NOT NULL,
JOB             VARCHAR2(9),
MGR             NUMBER(4),
HIREDATE        DATE DEFAULT                        SYSDATE,
SAL             NUMBER(7, 2),
COMM            NUMBER(7, 2),
DEPTNO          NUMBER(7, 2)                        NOT NULL,
CONSTRAINT      EMP_DEPTNO_CK CHECK (DEPTNO BETWEEN 1 AND 50),
CONSTRAINT      EMP_DEPTNO_FK                       FOREIGN KEY (DEPTNO)
REFERENCES      DEPT(DEPTNO)     );
```

Check constraint checks the value must be either equal or with in the provided range.

Foreign key constraint applies on a ***column of this table*** *Reference to a* ***particular column*** *of* ***another table.***

Constraints

Composite primary key

Example : Assume that we are making composite primary key on department table as:

```
create table dept(
dept_id number(8),
dept_name varchar2(30),
loc_id number(4),
constraint pk_dept primary key(dept_id,loc_id)  );
```

This is a ***composite primary key .*** *Key made from* ***dept_it*** *and* ***Loc_id***

Creating view

Creating Views

What is a view?

VIEW is a virtual table that does not physically exist. Rather, it is created by a query joining one or more tables.

Type of VIEWS

There are two type of view simple and complex views. The differnece between these two is as:

SIMPLE VIEWS	COMPLEX VIEWS
Views form by single table records	Views form by multiple table records
Do not contain functions in their queries	Contain functions in queries
Do not contain group by clause in it	May contain group by clause in it
DML operations can be performed via simple views which is not advisable	DML Operations can not be performed

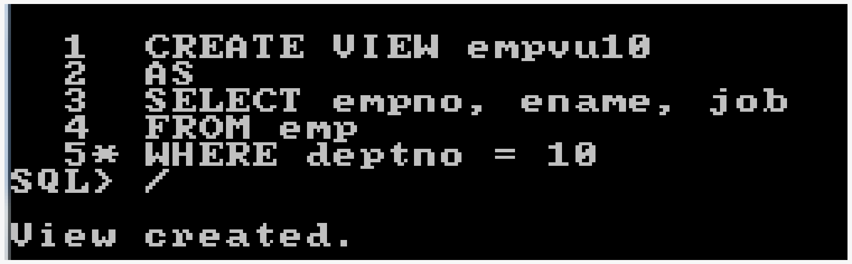
Creating Views

Syntax of VIEW

CREATE [OR REPLACE] VIEW_NAME view
AS
QUERY;

Example create a view named as **EMPVU10** that contains the employee number, name and job title for all the employees in department 10.(Rights required for create view in scott)

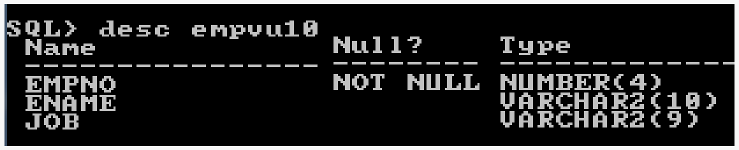
CREATE VIEW empvu10
AS
SELECT empno, ename, job
FROM emp
WHERE deptno = 10;



```
  1   CREATE VIEW empvu10
  2   AS
  3   SELECT empno, ename, job
  4   FROM emp
  5*  WHERE deptno = 10
SQL> /

View created.
```

We can display the structure of the view by using the SQL*Plus DESCRIBE command as

DESC empvu10



```
SQL> desc empvu10
 Name                            Null?    Type
 ------------------------------- -------- ------------
 EMPNO                           NOT NULL NUMBER(4)
 ENAME                                    VARCHAR2(10)
 JOB                                      VARCHAR2(9)
```

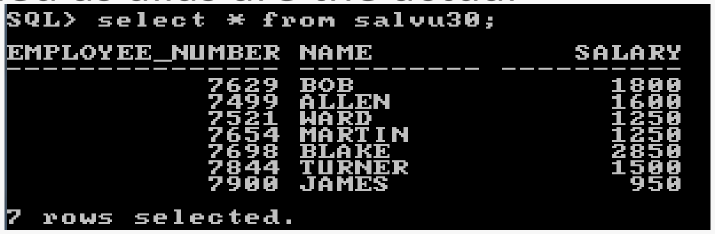
Creating Views

Creating views by using Column Aliases

```
CREATE VIEW salvu30
AS
SELECT empno EMPLOYEE_NUMBER, ename NAME,
sal SALARY
FROM emp
WHERE deptno = 30;
```

Now whatever the column names we have provided as alias are the actual column names of that view. You may test this as:

```
SELECT * FROM salvu30;
```



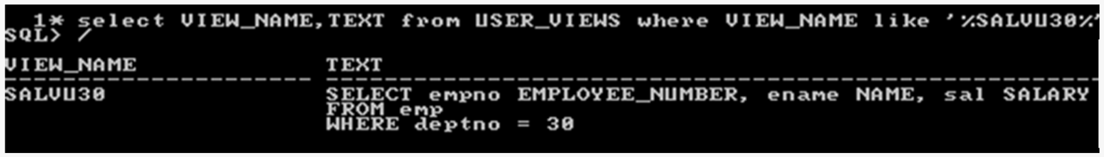
```
SQL> select * from salvu30;

EMPLOYEE_NUMBER NAME           SALARY
--------------- ---------- ----------
           7629 BOB              1800
           7499 ALLEN            1600
           7521 WARD             1250
           7654 MARTIN           1250
           7698 BLAKE            2850
           7844 TURNER           1500
           7900 JAMES             950

7 rows selected.
```

Views in the Data Dictionary

Once a view has been created, we can query the data dictionary table called USER_VIEWS to see the name of the view and the view definition. The text of the SELECT statement that constitutes the view is stored in a LONG column. You may check your newly create view via following query

Creating Views

Views in the Data Dictionary



```
  1* select VIEW_NAME,TEXT from USER_VIEWS where VIEW_NAME like '%SALVU30%'
SQL> /

VIEW_NAME                      TEXT
------------------------------ --------------------------------------------------
SALVU30                        SELECT empno EMPLOYEE_NUMBER, ename NAME, sal SALARY
                               FROM emp
                               WHERE deptno = 30
```

```
select VIEW_NAME,TEXT from USER_VIEWS
where VIEW_NAME like '%SALVU30%'
```

Creating a complex View

As already discussed that a complex view contains columns from multiple tables and may also include group functions.

Example : Create a view to show employee number, employee name and department name.

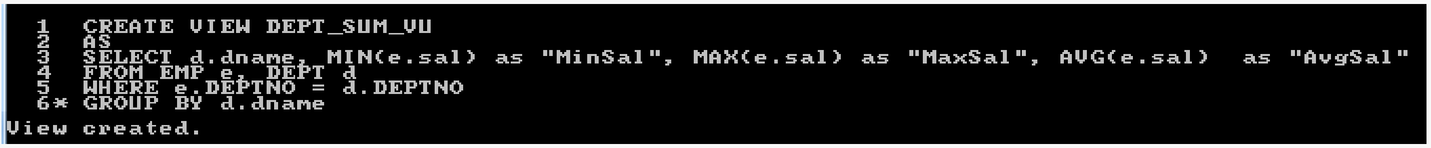
```
CREATE VIEW EMP_DEPT
AS
SELECT EMPNO, ENAME, DNAME
FROM EMP, DEPT
WHERE EMP.DEPTNO = DEPT.DEPTNO;
```

Creating Views

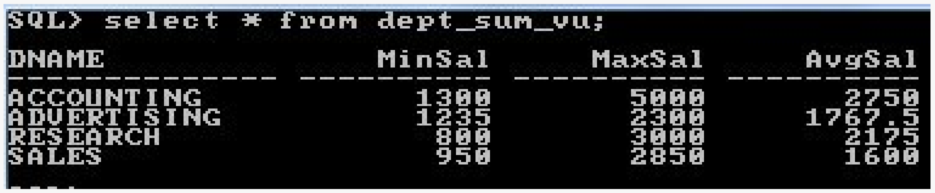
Creating a complex View

Example : Create a view that displays department wise Minimum, maximum and average salary for all employees

```
CREATE VIEW DEPT_SUM_VU
AS
SELECT d.dname, MIN(e.sal) as "MinSal", MAX(e.sal) as "MaxSal", AVG(e.sal)  as "AvgSal"
FROM EMP e, DEPT d
WHERE e.DEPTNO = d.DEPTNO
GROUP BY d.dname
```



```
  1  CREATE VIEW DEPT_SUM_VU
  2  AS
  3  SELECT d.dname, MIN(e.sal) as "MinSal", MAX(e.sal) as "MaxSal", AVG(e.sal)  as "AvgSal"
  4  FROM EMP e, DEPT d
  5  WHERE e.DEPTNO = d.DEPTNO
  6* GROUP BY d.dname
View created.
```



```
SQL> select * from dept_sum_vu;

DNAME              MinSal     MaxSal     AvgSal
-------------- ---------- ---------- ----------
ACCOUNTING           1300       5000       2750
ADVERTISING          1235       2300     1767.5
RESEARCH              800       3000       2175
SALES                 950       2850       1600
```

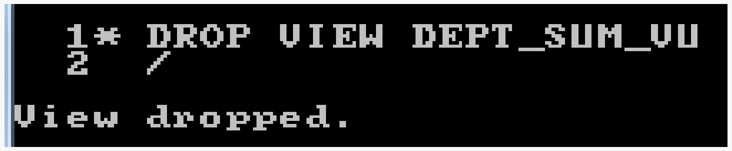
10

Removing View

Removing View

Just like the DROP TABLE, **DROP VIEW** statement removes the View. To remove the newly created **DEPT_SUM_VU** view we will write

DROP VIEW DEPT_SUM_VU



```
  1* DROP VIEW DEPT_SUM_VU
  2  /

View dropped.
```