Table of Contents

1. Data types for temporal data

Oracle has a few data types for temporal data

- Date: which stores both a date and a time with a precision of second
- TimeStamp: which stores both a date and a time with a precision of fractional seconds
- Interval Year To Month - this includes a precision for years of 0-9 digits with a default of 2 digits. A precision of 2 digits allows a maximum interval of 99 years and 11 months. The month component is integral.
- Interval Day To Second - this stores data as day, hours, minutes, seconds, and fractions of a second

The way that Oracle has implemented temporal intervals is not very intuitive and causes errors where it really shouldn't. I am not going to discuss time intervals in this class.
We don't really need the accuracy of TimeStamp for this class- so I only mention it briefly.
So we will focus on the Date data type.

2. How Date values are stored

The Oracle DATE data type stores values that include **a date and a time component** for each value. When Oracle stores a DATE value, it stores the value in a proprietary format as 7 components: Century, Year, Month, Day, Hour, Minute, Second. Oracle can handle dates from 4712 BC to 9999 CE and times from 0:00:00 to 23:59:59.

The earliest date for Oracle is January 1, 4712 BC.

3. How Date values are written and displayed

The people who develop a dbms and client software have to make some decisions about the default display for data. Oracle's decision for date values is to display the day using two digits, the month using three characters and the year using two digits. So if I do

```
select sysdate from dual;
```

I get back a display such as the following. The time component in that value for sysdate exists but it isn't displayed.

```
SYSDATE
---------
13-FEB-16
```

If you look at the inserts I gave you for the vets set of tables, the animal dob in the animals table was given using just the year-month-day part

```
insert into vt_animals ( an_id, cl_id, an_name, an_type, an_dob)
  values(15165, 411, 'Burgess', 'dog', date '2005-11-20' );
```

And the inserts used for the exam_date in the exam headers table had a time component.

```
insert into vt_exam_headers(ex_id, an_id, stf_id, ex_date)
  values (3105, 17002, 29, TO_DATE('2014-10-10 9:15'  , 'YYYY-MM-DD HH24:MI') );
```

And I had to do more work to specify that.

Both of the columns were simply defined as date type. The values for the an_dob have the time set to midnight. And I included a constraint that you could not use a different time component for the an_dob.

```
constraint an_dob_ck           check (trunc(an_dob) = an_dob)
```

Look at that constraint again after you read about the temporal functions.

If we display exam date values, we get the default display and the time component is not displayed.

```
select * from vt_exam_headers;
```

```
    EX_ID      AN_ID     STF_ID EX_DATE
---------- ---------- ---------- ---------
      2290      21320         38 11-APR-15
      2300      21316         38 08-MAY-15
      2352      10002         38 12-MAY-15
      2389      21006         38 20-MAY-15
      2400      12038         38 02-JUN-15
. . .
```

In the document on Conversion function in this unit I show you how to display dates in a variety of formats. In this document I will show you how to change the default date format for a session. (If you decide to do that for an assignment, you need to include that command in your script file.

4. TO_DATE and ANSI DATE literals

With Oracle, you can specify a date value with an ANSI date literal, such as Date '2003-08-15'. You need to use the format 'YYYY-MM-DD'. ANSI date literals do not include a time component.

Demo 01: The following and the next 4 demos create a table and insert 14 rows of date values.

```
create table z_tst_cal ( ID NUMBER(2), MyDate DATE);
```

Demo 02: -- Using the To_Date function

```
insert into z_tst_cal values  ( 1, TO_DATE('1/1/2003',    'MM/DD/YYYY') );
insert into z_tst_cal values  ( 2, TO_DATE('2/28/2003',   'MM/DD/YYYY') );
insert into z_tst_cal values  ( 3, TO_DATE('2003-05-11',  'YYYY-MM-DD') );
insert into z_tst_cal values  ( 4, TO_DATE('2003-jun-05', 'YYYY-MON-DD') );
insert into z_tst_cal values  ( 5, TO_DATE('20030908',    'YYYYMMDD') );
```

Demo 03: -- Specifying a time component

```
insert into z_tst_cal values  ( 6, TO_DATE('2003-08-19 1200pm', 'YYYY-MM-DD HHMIAM') );
insert into z_tst_cal values  ( 7, TO_DATE('2003-08-19 0600am', 'YYYY-MM-DD HHMIAM') );
insert into z_tst_cal values  ( 8, TO_DATE('2003-08-19 1200am', 'YYYY-MM-DD HHMIAM') );
insert into z_tst_cal values  ( 9, TO_DATE('2003-08-19 0600pm', 'YYYY-MM-DD HHMIAM') );
```

Demo 04: -- Specifying BC or AD

```
insert into z_tst_cal values  (10, TO_DATE('40000101BC', 'YYYYMMDDBC') );
insert into z_tst_cal values  (11, TO_DATE('99991231AD', 'YYYYMMDDBC') );
```

Demo 05: -- Using DATE

```
insert into z_tst_cal values  (12, DATE '2003-06-25' );
insert into z_tst_cal values  (13, DATE '2003-6-5' );
insert into z_tst_cal values  (14, DATE '03-06-25' );
```

Demo 06: This shows the values in the table, using two different display formats.

```
select ID
, TO_CHAR (MYDATE, 'YYYY-MM-DD A.D. HHMIAM')  AS  COL2
, TO_CHAR (MYDATE, 'DD MON, YYYY')  AS  COL3
from z_tst_cal;
```

```
         ID    COL2                      COL3
----------    ----------------------    ------------
          1    2003-01-01 A.D. 1200AM    01 JAN, 2003
          2    2003-02-28 A.D. 1200AM    28 FEB, 2003
          3    2003-05-11 A.D. 1200AM    11 MAY, 2003
          4    2003-06-05 A.D. 1200AM    05 JUN, 2003
          5    2003-09-08 A.D. 1200AM    08 SEP, 2003
          6    2003-08-19 A.D. 1200PM    19 AUG, 2003
          7    2003-08-19 A.D. 0600AM    19 AUG, 2003
          8    2003-08-19 A.D. 1200AM    19 AUG, 2003
          9    2003-08-19 A.D. 0600PM    19 AUG, 2003
         10    4000-01-01 B.C. 1200AM    01 JAN, 4000
         11    9999-12-31 A.D. 1200AM    31 DEC, 9999
         12    2003-06-25 A.D. 1200AM    25 JUN, 2003
         13    2003-06-05 A.D. 1200AM    05 JUN, 2003
         14    0003-06-25 A.D. 1200AM    25 JUN, 0003
```

5. Getting the current date

Sysdate returns the current date and time of the operating system on which the database resides. Current_date is the date and time of the user's session

Current_timestamp is another way to get the date and time.

Demo 07:

```
select sysdate  from dual;
```

```
SYSDATE
---------
13-FEB-16
```

```
select current_date  from dual;
```

```
CURRENT_D
---------
13-FEB-16
```

```
select current_timestamp from dual;
```

-- using the SQL *Plus client

```
CURRENT_TIMESTAMP
---------------------------------------------------------------------------
13-FEB-16 06.07.45.468000 PM -08:00
```

The display will be different in SQL Developer.

```
select to_char( sysdate, 'YYYY-MM-DD HHMIAM')  from dual;
```

```
TO_CHAR(SYSDATE,'
-----------------
2016-02-13 0609PM
```

```
select to_char( current_date, 'YYYY-MM-DD HHMIAM')  from dual;
```

```
TO_CHAR(CURRENT_D
-----------------
2016-02-13 0609PM
```

6. TimeStamp data

The timestamp data type will store date values to fractional parts of a second, with a precision of 0 to 9 digits. Sysdate returns the system date as a date type; to get a current time as a timestamp value use SysTimeStamp.

Example of the Timestamp data type.

Demo 08: If you run this, copy and paste the first two inserts as a pair. Then do the other inserts one at a time rather than paste them all at once. Maybe go get a cup of coffee between some of the inserts.

```
create table z_tst_timeStamp (id number, simple_date DATE, time_stamp_date TIMESTAMP);

insert into  z_tst_timeStamp values (1, SYSDATE, systimestamp);
insert into  z_tst_timeStamp values (2, SYSDATE, systimestamp);

insert into  z_tst_timeStamp values (3, SYSDATE, systimestamp);

insert into  z_tst_timeStamp values (4, SYSDATE, systimestamp);
commit;
```

Demo 09: sample table data

```
select ID
, to_char(simple_date, 'YYYY-MM-DD HHMIAM') as "Simple Date"
, time_stamp_date as "TimeStamp Date"
from z_tst_timeStamp;
```

```
       ID Simple Date       TimeStamp Date
--------- ----------------- ------------------------------
        1 2016-02-13 0611PM 13-FEB-16 06.11.06.104000 PM
        2 2016-02-13 0611PM 13-FEB-16 06.11.06.125000 PM
        3 2016-02-13 0611PM 13-FEB-16 06.11.46.826000 PM
        4 2016-02-13 0612PM 13-FEB-16 06.12.30.071000 PM
```

In this case the first rows have the same value for the Simple Date column within a minute. The timestamp values are different even with a copy and paste of the two inserts. Timestamp is a more precise type.

7. Assumptions we make and why we shouldn't

This is semi-optional- which means I want you to read this and think about it- but it is not on the test or assignments.

Since it is assumed for this class that we are all working on the same Oracle system and that it has been set up with certain defaults, we are working under certain assumptions. And for this class, that is OK. But we should at least mention some issues in this area.

Demo 10: Suppose you want to get all the animals that were born in the month of March. Trust me- this is a terrible idea but you will see it in a lot of old sql.

```
select an_id, an_dob
from vt_animals
where an_dob like '%MAR%';
```

```
    AN_ID AN_DOB
---------- ---------
     15401 15-MAR-10
     21005 06-MAR-11
     21205 30-MAR-15
```

Demo 11: Or you want to find animals born after the 15th day of any month. This is also a terrible way to do this.

```
select an_id, an_dob
from vt_animals
where substr(an_dob, 1, 2) > 15;
```

```
     AN_ID AN_DOB
---------- ---------
     11015 23-FEB-12
     12035 28-FEB-95
     12038 29-APR-12
     15165 20-NOV-05
     16002 25-MAY-15
     21001 22-MAY-09
     21003 18-JUN-14
     21205 30-MAR-15
     21305 27-JUL-14
     21306 27-JUL-14
     21307 27-JUL-14

 11 rows selected
```

These seem to work and seem pretty clever.

But then someone changes the default date format to match the standard- you can do this on a session basis or the dba can do this for the system.

Demo 12:

```
alter session set nls_date_format= 'YYYY-MM-DD';
Session altered
```

If you run those two queries again- it looks like no animals were born in March and a lot more animals were born after the 15th of the month.

```
SQL> select an_id, an_dob
  2  from vt_animals
  3  where an_dob like '%MAR%';

no rows selected

SQL> select an_id, an_dob
  2  from vt_animals
  3  where substr(an_dob, 1, 2) > 15;

     AN_ID AN_DOB
---------- ----------
     10002 2010-04-15
     11015 2012-02-23
     11025 2012-02-01
     11028 2015-10-01
     11029 2015-10-01

38 rows selected.
```

What those queries are doing is treating the date value as a string and then doing string manipulation. It is better to use functions which are designed to handle dates to do date manipulation. (See the next document for how to do these correctly.)

Date values are not strings; don't pretend they are strings

The result of a To_CHAR function is a string.

You can reset the nls_date_format back - or just log out and start a new session.

```
alter session set nls_date_format='DD-MON-YY';
```