16

Oracle9*i* Datetime Functions

ORACLE

Schedule:	Timing	Topic
	30 minutes	Lecture
	20 minutes	Practice
	50 minutes	Total

Objectives

After completing this lesson, you should be able use the following datetime functions:

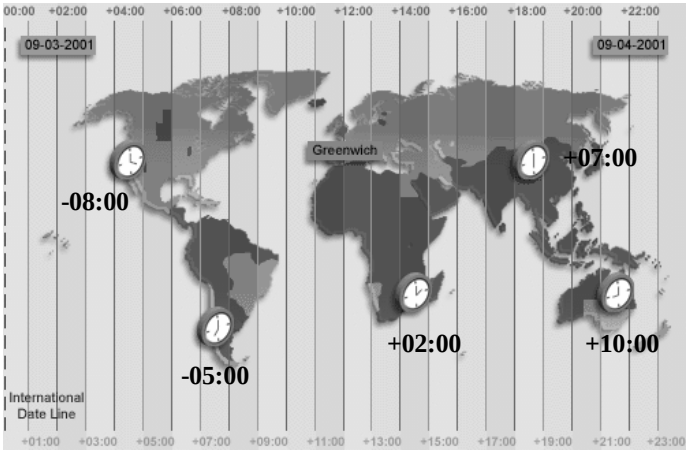
- **TZ_OFFSET**
- **CURRENT_DATE**
- **CURRENT_TIMESTAMP**
- **LOCALTIMESTAMP**
- **DBTIMEZONE**
- **SESSIONTIMEZONE**
- **EXTRACT**
- **FROM_TZ**
- **TO_TIMESTAMP**
- **TO_TIMESTAMP_TZ**
- **TO_YMINTERVAL**

ORACLE

Lesson Aim

This lesson addresses some of the datetime functions introduced in Oracle9*i*.

TIME ZONES



The image represents the time for each time zone when Greenwich time is 12:00.

ORACLE

Time Zones

In Oracle9*i*, you can include the time zone in your date and time data, as well as provide support for fractional seconds. This lesson focuses on how to manipulate the new datetime data types included with Oracle9*i* using the new datetime functions. To understand the working of these functions, it is necessary to be familiar with the concept of time zones and Greenwich Mean Time, or GMT. Greenwich Mean Time, or GMT is now referred to as UTC (Coordinated Universal Time).

The hours of the day are measured by the turning of the earth. The time of day at any particular moment depends on where you are. When it is noon in Greenwich, England, it is midnight along the international date line. The earth is divided into 24 time zones, one for each hour of the day. The time along the prime meridian in Greenwich, England is known as Greenwich mean time, or GMT. GMT is the time standard against which all other time zones in the world are referenced. It is the same all year round and is not effected by summer time or daylight savings time. The meridian line is an imaginary line that runs from the North Pole to the South Pole. It is known as zero longitude and it is the line from which all other lines of longitude are measured. All time is measured relative to Greenwich mean time (GMT) and all places have a latitude (their distance north or south of the equator) and a longitude (their distance east or west of the Greenwich meridian).

Daylight Saving Time

Most western nations advance the clock ahead one hour during the summer months. This period is called daylight saving time. Daylight saving time lasts from the first Sunday in April to the last Sunday in October in the most of the United States, Mexico and Canada. The nations of the European Union observe daylight saving time, but they call it the summer time period. Europe's summer time period begins a week earlier than its North American counterpart, but ends at the

Oracle9*i* Datetime Support

- **In Oracle9*i*, you can include the time zone in your date and time data, and provide support for fractional seconds.**
- **Three new data types are added to `DATE`:**
  - **`TIMESTAMP`**
  - **`TIMESTAMP WITH TIME ZONE (TSTZ)`**
  - **`TIMESTAMP WITH LOCAL TIME ZONE (TSLTZ)`**
- **Oracle9*i* provides daylight savings support for datetime data types in the server.**

ORACLE

Oracle9*i* Datetime Support

With Oracle9*i*, three new data types are added to `DATE`, with the following differences:

Data Type	Time Zone	Fractional Seconds
`DATE`	No	No
`TIMESTAMP`	No	Yes
`TIMESTAMP (`*`fractional_seconds_precision`*`) WITH TIMEZONE`	All values of `TIMESTAMP` as well as the time zone displacement value which indicates the hours and minutes before or after UTC (Coordinated Universal Time, formerly Greenwich mean time).	*`fractional_seconds_precision`* is the number of digits in the fractional part of the `SECOND` datetime field. Accepted values are 0 to 9. The default is 6.
`TIMESTAMP (`*`fractional_seconds_precision`*`) WITH LOCAL TIME ZONE`	All values of `TIMESTAMP WITH TIME ZONE`, with the following exceptions: • Data is normalized to the database time zone when it is stored in the database. • When the data is retrieved, users see the data in the session time zone.	Yes

`TIMESTAMP WITH LOCAL TIME ZONE` is stored in the database time zone. When a user selects the data, the value is adjusted to the user's session time zone.

Example:

A San Francisco database has system time zone = -8:00. When a New York client (session time zone = -5:00) inserts into or selects from the San Francisco database, `TIMESTAMP WITH LOCAL TIME ZONE` data is adjusted as follows:

- The New York client inserts `TIMESTAMP '1998-1-23 6:00:00-5:00'` into a `TIMESTAMP WITH LOCAL TIME ZONE` column in the San Francisco database. The inserted data is stored in San Francisco as binary value `1998-1-23 3:00:00`.
- When the New York client selects that inserted data from the San Francisco database, the value displayed in New York is `'1998-1-23 6:00:00'`.
- A San Francisco client, selecting the same data, see the value `'1998-1-23 3:00:00'`.

Support for Daylight Savings Times

The Oracle Server automatically determines, for any given time zone region, whether daylight savings is in effect and returns local time values based accordingly. The datetime value is sufficient for the server to determine whether daylight savings time is in effect for a given region in all cases except boundary cases. A boundary case occurs during the period when daylight savings goes into or comes out of effect. For example, in the U.S.-Pacific region, when daylight savings comes into effect, the time changes from 2:00 a.m. to 3:00 a.m. The one hour interval between 2:00 a.m. and 3:00 a.m. does not exist. When daylight savings goes out of effect, the time changes from 2:00 a.m. back to 1:00 a.m., and the one-hour interval between 1:00 a.m. and 2:00 a.m. is repeated.

Oracle9*i* also significantly reduces the cost of developing and deploying applications globally on a single database instance. Requirements for multigeographic applications include named time zones and multilanguage support through Unicode. The datetime data types `TSLTZ` and `TSTZ` are time-zone-aware. Datetime values can be specified as local time in a particular region (rather than a particular offset). Using the time zone rules tables for a given region, the time zone offset for a local time is calculated, taking into consideration daylight savings time adjustments, and used in further operations.

This lesson addresses some of the new datetime functions introduced in Oracle9*i*.

TZ_OFFSET

- **Display the time zone offset for the time zone `'US/Eastern'`**

```
SELECT TZ_OFFSET('US/Eastern') FROM DUAL;
```

TZ_OFFS
-04:00

- **Display the time zone offset for the time zone `'Canada/Yukon'`**

```
SELECT TZ_OFFSET('Canada/Yukon') FROM DUAL;
```

TZ_OFFS
-07:00

- **Display the time zone offset for the time zone `'Europe/London'`**

```
SELECT TZ_OFFSET('Europe/London') FROM DUAL;
```

TZ_OFFS
+01:00

ORACLE

TZ_OFFSET

The TZ_OFFSET function returns the time zone offset corresponding to the value entered. The return value is dependent on the date when the statement is executed. For example if the TZ_OFFSET function returns a value -08:00, the return value can be interpreted as the time zone from where the command was executed is eight hours after UTC. You can enter a valid time zone name, a time zone offset from UTC (which simply returns itself), or the keyword SESSIONTIMEZONE or DBTIMEZONE. The syntax of the TZ_OFFSET function is:

```
TZ_OFFSET ( ['time_zone_name'] '[+ | -] hh:mm']
            [ SESSIONTIMEZONE] [DBTIMEZONE])
```

The examples in the slide can be interpreted as follows:

- The time zone `'US/Eastern'` is four hours behind UTC
- The time zone `'Canada/Yukon'` is seven hours behind UTC
- The time zone `'Europe/London'` is one hour ahead of UTC

For a listing of valid time zone name values, query the V$TIMEZONE_NAMES dynamic performance view.

```
DESC V$TIMEZONE_NAMES
```

Name	Null?	Type
TZNAME		VARCHAR2(64)
TZABBREV		VARCHAR2(64)

```
SELECT * FROM  V$TIMEZONE_NAMES;
```

TZNAME	TZABBREV
Africa/Cairo	LMT
Africa/Cairo	EET
Africa/Cairo	EEST
Africa/Tripoli	LMT
Africa/Tripoli	CET
Africa/Tripoli	CEST
Africa/Tripoli	EET
America/Adak	LMT
America/Adak	NST
America/Adak	NWT
America/Adak	BST
America/Adak	BDT
America/Adak	HAST
America/Adak	HADT
TZNAME	TZABBREV
America/Anchorage	LMT
America/Anchorage	CAT
America/Anchorage	CAWT
America/Anchorage	AHST
America/Anchorage	AHDT
America/Anchorage	AKST
...	
W-SU	MDST
W-SU	S
W-SU	MSD
W-SU	MSK
W-SU	EET
W-SU	EEST
WET	WEST
WET	WET

616 rows selected.

CURRENT_DATE

- **Display the current date and time in the session's time zone .**

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
```

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_DATE
-05:00	03-OCT-2001 09:37:06

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_DATE
-08:00	03-OCT-2001 06:38:07

- **CURRENT_DATE is sensitive to the session time zone.**
- **The return value is a date in the Gregorian calendar.**

ORACLE

CURRENT_DATE

The CURRENT_DATE function returns the current date in the session's time zone. The return value is a date in the Gregorian calendar.

The examples in the slide illustrate that CURRENT_DATE is sensitive to the session time zone. In the first example, the session is altered to set the TIME_ZONE parameter to –5:0. The TIME_ZONE parameter specifies the default local time zone displacement for the current SQL session. TIME_ZONE is a session parameter only, not an initialization parameter. The TIME_ZONE parameter is set as follows:

```
TIME_ZONE = '[+ | -] hh:mm'
```

The format mask ([+ | -] hh:mm) indicates the hours and minutes before or after UTC (Coordinated Universal Time, formerly known as Greenwich mean time).

Observe in the output that the value of CURRENT_DATE changes when the TIME_ZONE parameter value is changed to –8:0 in the second example.

Note: The ALTER SESSION command sets the date format of the session to 'DD-MON-YYYY HH24:MI:SS' that is Day of month (1-31)-Abbreviated name of month-4-digit year Hour of day (0-23):Minute (0-59):Second (0-59).

Instructor Note

You might also want to select the SYSDATE for each TIME_ZONE and draw the attention of the students to the fact that SYSDATE remains the same irrespective of the change in the TIME_ZONE.
SYSDATE is not sensitive to the session's time zone.

CURRENT_TIMESTAMP

- **Display the current date and fractional time in the session's time zone.**

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-05:00	03-OCT-01 09.40.59.000000 AM -05:00

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP
FROM DUAL;
```

SESSIONTIMEZONE	CURRENT_TIMESTAMP
-08:00	03-OCT-01 06.41.38.000000 AM -08:00

- **CURRENT_TIMESTAMP is sensitive to the session time zone.**
- **The return value is of the TIMESTAMP WITH TIME ZONE datatype.**

CURRENT_TIMESTAMP

The CURRENT_TIMESTAMP function returns the current date and time in the session time zone, as a value of the data type TIMESTAMP WITH TIME ZONE. The time zone displacement reflects the current local time of the SQL session. The syntax of the CURRENT_TIMESTAMP function is:

```
CURRENT_TIMESTAMP (precision)
```

where *precision* is an optional argument that specifies the fractional second precision of the time value returned. If you omit precision, the default is 6.

The examples in the slide illustrates that CURRENT_TIMESTAMP is sensitive to the session time zone. In the first example, the session is altered to set the TIME_ZONE parameter to –5:0. Observe in the output that the value of CURRENT_TIMESTAMP changes when the TIME_ZONE parameter value is changed to –8:0 in the second example.

LOCALTIMESTAMP

- **Display the current date and time in the session time zone in a value of TIMESTAMP data type.**

```
ALTER SESSION SET TIME_ZONE = '-5:0';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

CURRENT_TIMESTAMP	LOCALTIMESTAMP
03-OCT-01 09.44.21.000000 AM -05:00	03-OCT-01 09.44.21.000000 AM

```
ALTER SESSION SET TIME_ZONE = '-8:0';
SELECT CURRENT_TIMESTAMP, LOCALTIMESTAMP
FROM DUAL;
```

CURRENT_TIMESTAMP	LOCALTIMESTAMP
03-OCT-01 06.45.21.000001 AM -08:00	03-OCT-01 06.45.21.000001 AM

- **LOCALTIMESTAMP returns a TIMESTAMP value, whereas CURRENT_TIMESTAMP returns a TIMESTAMP WITH TIME ZONE value.**

ORACLE

LOCALTIMESTAMP

The LOCALTIMESTAMP function returns the current date and time in the session time zone in a value of data type TIMESTAMP. The difference between this function and CURRENT_TIMESTAMP is that LOCALTIMESTAMP returns a TIMESTAMP value, while CURRENT_TIMESTAMP returns a TIMESTAMP WITH TIME ZONE value. TIMESTAMP WITH TIME ZONE is a variant of TIMESTAMP that includes a time zone displacement in its value. The time zone displacement is the difference (in hours and minutes) between local time and UTC. The TIMESTAMP WITH TIME ZONE data type has the following format:

```
TIMESTAMP [ (fractional_seconds_precision) ] WITH TIME ZONE
```

where *fractional_seconds_precision* optionally specifies the number of digits in the fractional part of the SECOND datetime field and can be a number in the range 0 to 9. The default is 6. For example, you specify TIMESTAMP WITH TIME ZONE as a literal as follows:

```
TIMESTAMP '1997-01-31 09:26:56.66 +02:00'
```

The syntax of the LOCAL_TIMESTAMP function is:

```
LOCAL_TIMESTAMP (TIMESTAMP_precision)
```

Where, *TIMESTAMP_precision* is an optional argument that specifies the fractional second precision of the TIMESTAMP value returned.

The examples in the slide illustrates the difference between LOCALTIMESTAMP and CURRENT_TIMESTAMP. Observe that the LOCALTIMESTAMP does not display the time zone value, while the CURRENT_TIMESTAMP does.

DBTIMEZONE and SESSIONTIMEZONE

- **Display the value of the database time zone.**

```
SELECT DBTIMEZONE FROM DUAL;
```

DBTIME
-05:00

- **Display the value of the session's time zone.**

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

SESSIONTIMEZONE
-08:00

ORACLE

DBTIMEZONE and SESSIONTIMEZONE

The default database time zone is the same as the operating system's time zone. You set the database's default time zone by specifying the SET TIME_ZONE clause of the CREATE DATABASE statement. If omitted, the default database time zone is the operating system time zone. The database time zone can be changed for a session with an ALTER SESSION statement.

The DBTIMEZONE function returns the value of the database time zone. The return type is a time zone offset (a character type in the format '[+|-]TZH:TZM') or a time zone region name, depending on how the user specified the database time zone value in the most recent CREATE DATABASE or ALTER DATABASE statement. The example on the slide shows that the database time zone is set to UTC, as the TIME_ZONE parameter is in the format:

```
TIME_ZONE = '[+ | -] hh:mm'
```

The SESSIONTIMEZONE function returns the value of the current session's time zone. The return type is a time zone offset (a character type in the format '[+|]TZH:TZM') or a time zone region name, depending on how the user specified the session time zone value in the most recent ALTER SESSION statement. The example in the slide shows that the session time zone is set to UTC.

Observe that the database time zone is different from the current session's time zone.

EXTRACT

- **Display the YEAR component from the SYSDATE.**

```
SELECT EXTRACT (YEAR FROM SYSDATE) FROM DUAL;
```

EXTRACT(YEARFROMSYSDATE)
2001

- **Display the MONTH component from the HIRE_DATE for those employees whose MANAGER_ID is 100.**

```
SELECT last_name, hire_date,
       EXTRACT (MONTH FROM HIRE_DATE)
FROM employees
WHERE manager_id = 100;
```

LAST_NAME	HIRE_DATE	EXTRACT(MONTHFROMHIRE_DATE)
Kochhar	21-SEP-89	9
De Haan	13-JAN-93	1
Mourgos	16-NOV-99	11
Zlotkey	29-JAN-00	1
Hartstein	17-FEB-96	2

ORACLE

EXTRACT

The EXTRACT expression extracts and returns the value of a specified datetime field from a datetime or interval value expression. You can extract any of the components mentioned in the following syntax using the EXTRACT function. The syntax of the EXTRACT function is:

```
SELECT  EXTRACT ([YEAR] [MONTH][DAY] [HOUR] [MINUTE][SECOND]
                 [TIMEZONE_HOUR] [TIMEZONE_MINUTE]
                 [TIMEZONE_REGION] [TIMEZONE_ABBR]
FROM    [datetime_value_expression]
        [interval_value_expression]);
```

When you extract a TIMEZONE_REGION or TIMEZONE_ABBR (abbreviation), the value returned is a string containing the appropriate time zone name or abbreviation. When you extract any of the other values, the value returned is in the Gregorian calendar. When extracting from a datetime with a time zone value, the value returned is in UTC. For a listing of time zone names and their corresponding abbreviations, query the V$TIMEZONE_NAMES dynamic performance view. In the first example on the slide, the EXTRACT function is used to extract the YEAR from SYSDATE.

In the second example in the slide, the EXTRACT function is used to extract the MONTH from HIRE_DATE column of the EMPLOYEES table, for those employees who report to the manager whose EMPLOYEE_ID is 100.

Instructor Note

The Oracle Server lets you derive datetime and interval value expressions. Datetime value expressions yield values of datetime data type. Interval value expressions yield values of interval data type. For more information on these data types refer *Oracle9i SQL Reference.*

TIMESTAMP Conversion Using FROM_TZ

- **Display the TIMESTAMP value '2000-03-28 08:00:00' as a TIMESTAMP WITH TIME ZONE value.**

```
SELECT FROM_TZ(TIMESTAMP
       '2000-03-28 08:00:00','3:00')
FROM DUAL;
```

FROM_TZ(TIMESTAMP'2000-03-2808:00:00','3:00')
28-MAR-00 08.00.00.000000000 AM +03:00

- **Display the TIMESTAMP value '2000-03-28 08:00:00' as a TIMESTAMP WITH TIME ZONE value for the time zone region 'Australia/North'**

```
SELECT FROM_TZ(TIMESTAMP
       '2000-03-28 08:00:00', 'Australia/North')
FROM DUAL;
```

FROM_TZ(TIMESTAMP'2000-03-2808:00:00','AUSTRALIA/NORTH')
28-MAR-00 08.00.00.000000000 AM AUSTRALIA/NORTH

ORACLE

TIMESTAMP Conversion Using FROM_TZ

The FROM_TZ function converts a TIMESTAMP value to a TIMESTAMP WITH TIME ZONE value.

The syntax of the FROM_TZ function is as follows:

```
FROM_TZ(TIMESTAMP  timestamp_value, time_zone_value)
```

where *time_zone_value* is a character string in the format 'TZH:TZM' or a character expression that returns a string in TZR (time zone region) with optional TZD format (TZD is an abbreviated time zone string with daylight savings information.) TZR represents the time zone region in datetime input strings. Examples are 'Australia/North', 'UTC', and 'Singapore'. TZD represents an abbreviated form of the time zone region with daylight savings information. Examples are 'PST' for US/Pacific standard time and 'PDT' for US/Pacific daylight time. To see a listing of valid values for the TZR and TZD format elements, query the V$TIMEZONE_NAMES dynamic performance view.

The example in the slide converts a TIMESTAMP value to TIMESTAMP WITH TIME ZONE.

TO_TIMESTAMP and TO_TIMESTAMP_TZ

- **Display the character string `'2000-12-01 11:00:00'` as a `TIMESTAMP` value.**

```
SELECT TO_TIMESTAMP ('2000-12-01 11:00:00',
                     'YYYY-MM-DD HH:MI:SS')
FROM DUAL;
```

TO_TIMESTAMP('2000-12-0111:00:00','YYYY-MM-DDHH:MI:SS')
01-DEC-00 11.00.00.000000000 AM

- **Display the character string `'1999-12-01 11:00:00 -8:00'` as a `TIMESTAMP WITH TIME ZONE` value.**

```
SELECT
  TO_TIMESTAMP_TZ('1999-12-01 11:00:00 -8:00',
                  'YYYY-MM-DD HH:MI:SS TZH:TZM')
FROM DUAL;
```

TO_TIMESTAMP_TZ('1999-12-0111:00:00-8:00','YYYY-MM-DDHH:MI:SSTZH:TZM')
01-DEC-99 11.00.00.000000000 AM -08:00

ORACLE

STRING To TIMESTAMP Conversion Using TO_TIMESTAMP and TO_TIMESTAMP_TZ

The TO_TIMESTAMP function converts a string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of TIMESTAMP data type. The syntax of the TO_TIMESTAMP function is:

```
TO_TIMESTAMP (char,[fmt],['nlsparam'])
```

The optional *fmt* specifies the format of *char*. If you omit *fmt*, the string must be in the default format of the TIMESTAMP data type. The optional *nlsparam* specifies the language in which month and day names and abbreviations are returned. This argument can have this form:

```
'NLS_DATE_LANGUAGE = language'
```

If you omit *nlsparams*, this function uses the default date language for your session. The example on the slide converts a character string to a value of TIMESTAMP.

The TO_TIMESTAMP_TZ function converts a string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to a value of TIMESTAMP WITH TIME ZONE data type. The syntax of the TO_TIMESTAMP_TZ function is:

```
TO_TIMESTAMP_TZ (char,[fmt],['nlsparam'])
```

The optional *fmt* specifies the format of char. If omitted, a string must be in the default format of the TIMESTAMP WITH TIME ZONE data type. The optional *nlsparam* has the same purpose in this function as in the TO_TIMESTAMP function. The example in the slide converts a character string to a value of TIMESTAMP WITH TIME ZONE.

Note: The TO_TIMESTAMP_TZ function does not convert character strings to TIMESTAMP WITH LOCAL TIME ZONE.

Time Interval Conversion with TO_YMINTERVAL

- **Display a date that is one year two months after the hire date for the employees working in the department with the DEPARTMENT_ID 20**

```
SELECT hire_date,
       hire_date + TO_YMINTERVAL('01-02') AS
       HIRE_DATE_YMININTERVAL
FROM EMPLOYEES
WHERE department_id = 20;
```

HIRE_DATE	HIRE_DATE_YMININTERV
17-FEB-1996 00:00:00	17-APR-1997 00:00:00
17-AUG-1997 00:00:00	17-OCT-1998 00:00:00

ORACLE

Time Interval Conversion with TO_YMINTERVAL

The TO_YMINTERVAL function converts a character string of CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to an INTERVAL YEAR TO MONTH data type. The INTERVAL YEAR TO MONTH data type stores a period of time using the YEAR and MONTH datetime fields. The format of INTERVAL YEAR TO MONTH is as follows:

```
INTERVAL YEAR [(year_precision)] TO MONTH
```

where *year_precision* is the number of digits in the YEAR datetime field. The default value of year_precision is 2.

The syntax of the TO_YMINTERVAL function is:

```
TO_YMINTERVAL (char)
```

where *char* is the character string to be converted.

The example in the slide calculates a date that is one year two months after the hire date for the employees working in the department 20 of the EMPLOYEES table.

A reverse calculation can also be done using the TO_YMINTERVAL function. For example:

```
SELECT hire_date, hire_date +  TO_YMINTERVAL('-02-04') AS
       HIRE_DATE_YMINTERVAL
FROM   employees WHERE department_id = 20;
```

Observe that the character string passed to the TO_YMINTERVAL function has a negative value. The example returns a date that is two years and four months before the hire date for the employees working in the department 20 of the EMPLOYEES table.

Summary

In this lesson, you should have learned how to use the following functions:

- **TZ_OFFSET**
- **FROM_TZ**
- **TO_TIMESTAMP**
- **TO_TIMESTAMP_TZ**
- **TO_YMINTERVAL**
- **CURRENT_DATE**
- **CURRENT_TIMESTAMP**
- **LOCALTIMESTAMP**
- **DBTIMEZONE**
- **SESSIONTIMEZONE**
- **EXTRACT**

ORACLE

Summary

This lesson addressed some of the new datetime functions introduced in Oracle9*i*.

Practice 16 Overview

This practice covers using the Oracle9*i* datetime functions.

ORACLE

Practice 16 Overview

In this practice, you display time zone offsets, `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and the `LOCALTIMESTAMP`. You also set time zones and use the `EXTRACT` function.

Instructor Note

1. If you have demonstrated the code example: `ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS'`, remember to issue `ALTER SESSION SET NLS_DATE_FORMAT = 'DD-MON-YYYY'` before moving on to the next lesson.

2. You might want to mention that the results of the questions are based on a different date, and in some cases they will not match the actual results that the students will get. Also, the time zone offset of the various countries might differ based on daylight saving time.

1. Alter the session to set the NLS_DATE_FORMAT to DD-MON-YYYY HH24:MI:SS.
2. a. Write queries to display the time zone offsets (TZ_OFFSET), for the following time zones.
   - *US/Pacific-New*

TZ_OFFS
-07:00

Singapore

TZ_OFFS
+08:00

TZ_OFFS
+02:00

b. Alter the session to set the TIME_ZONE parameter value to the time zone offset of US/Pacific-New.

c. Display the CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session.

Note: The output might be different based on the date when the command is executed.

CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
01-OCT-2001 13:40:54	01-OCT-01 01.40.54.000001 PM -07:00	01-OCT-01 01.40.54.000001 PM

Singapore.

e. Display the CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP for this session. Note: The output might be different based on the date when the command is executed.

CURRENT_DATE	CURRENT_TIMESTAMP	LOCALTIMESTAMP
02-OCT-2001 04:42:34	02-OCT-01 04.42.34.000000 AM +08:00	02-OCT-01 04.42.34.000000 AM

LOCALTIMESTAMP are all sensitive to the session time zone.

3. Write a query to display the DBTIMEZONE and SESSIONTIMEZONE.

DBTIME	SESSIONTIMEZONE
-05:00	+08:00

Practice 2: (Continued)

4. Write a query to extract the YEAR from HIRE_DATE column of the EMPLOYEES table for those
employees who work in department 80.

LAST_NAME	EXTRACT(YEARFROMHIRE_DATE)
Zlotkey	2000
Abel	1996
Taylor	1998

5. Alter the session to set the NLS_DATE_FORMAT to DD-MON-YYYY.