

10

Managing Data in Different Time Zones

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Objectives

After completing this lesson, you should be able to:

- Use data types similar to `DATE` that store fractional seconds and track time zones
- Use data types that store the difference between two datetime values
- Use the following datetime functions:
 - `CURRENT_DATE`
 - `CURRENT_TIMESTAMP`
 - `LOCALTIMESTAMP`
 - `DBTIMEZONE`
 - `SESSIONTIMEZONE`
 - `EXTRACT`
 - `TZ_OFFSET`
 - `FROM_TZ`
 - `TO_TIMESTAMP`
 - `TO_YMINTERVAL`
 - `TO_DSINTERVAL`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this lesson, you learn how to use data types similar to `DATE` that store fractional seconds and track time zones. This lesson addresses some of the datetime functions available in the Oracle database.

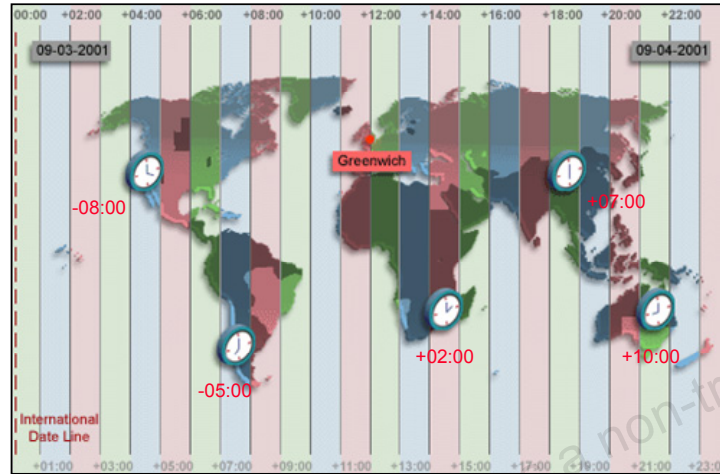
Lesson Agenda

- **CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP**
- INTERVAL data types
- Using the following functions:
 - EXTRACT
 - TZ_OFFSET
 - FROM_TZ
 - TO_TIMESTAMP
 - TO_YMINTERVAL
 - TO_DSINTERVAL

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Time Zones



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

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 (GMT). GMT is now known as Coordinated Universal Time (UTC). UTC 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 affected by summer time or daylight saving 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 UTC 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).

TIME_ZONE Session Parameter

TIME_ZONE may be set to:

- An absolute offset
- Database time zone
- OS local time zone
- A named region

```
ALTER SESSION SET TIME_ZONE = '-05:00';  
ALTER SESSION SET TIME_ZONE = dbtimezone;  
ALTER SESSION SET TIME_ZONE = local;  
ALTER SESSION SET TIME_ZONE = 'America/New_York';
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The Oracle database supports storing the time zone in your date and time data, as well as fractional seconds. The `ALTER SESSION` command can be used to change time zone values in a user's session. The time zone values can be set to an absolute offset, a named time zone, a database time zone, or the local time zone.

CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP

- **CURRENT_DATE:**
 - Returns the current date from the user session
 - Has a data type of DATE
- **CURRENT_TIMESTAMP:**
 - Returns the current date and time from the user session
 - Has a data type of TIMESTAMP WITH TIME ZONE
- **LOCALTIMESTAMP:**
 - Returns the current date and time from the user session
 - Has a data type of TIMESTAMP

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The **CURRENT_DATE** and **CURRENT_TIMESTAMP** functions return the current date and current time stamp, respectively. The data type of **CURRENT_DATE** is DATE. The data type of **CURRENT_TIMESTAMP** is **TIMESTAMP WITH TIME ZONE**. The values returned display the time zone displacement of the SQL session executing the functions. 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 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 through 9. The default is 6.

The **LOCALTIMESTAMP** function returns the current date and time in the session time zone. The difference between **LOCALTIMESTAMP** and **CURRENT_TIMESTAMP** is that **LOCALTIMESTAMP** returns a **TIMESTAMP** value, whereas **CURRENT_TIMESTAMP** returns a **TIMESTAMP WITH TIME ZONE** value.

These functions are national language support (NLS)–sensitive—that is, the results will be in the current NLS calendar and datetime formats.

Note: The **SYSDATE** function returns the current date and time as a **DATE** data type. You learned how to use the **SYSDATE** function in the course titled *Oracle Database: SQL Workshop I*.

Comparing Date and Time in a Session's Time Zone

The `TIME_ZONE` parameter is set to `-5:00` and then `SELECT` statements for each date and time are executed to compare differences.

```
ALTER SESSION
SET NLS_DATE_FORMAT = 'DD-MON-YYYY HH24:MI:SS';
ALTER SESSION SET TIME_ZONE = '-5:00';

SELECT SESSIONTIMEZONE, CURRENT_DATE FROM DUAL; 1

SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL; 2

SELECT SESSIONTIMEZONE, LOCALTIMESTAMP FROM DUAL; 3
```

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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).

The example in the slide illustrates that the session is altered to set the `TIME_ZONE` parameter to `-5:00`. Then the `SELECT` statement for `CURRENT_DATE`, `CURRENT_TIMESTAMP`, and `LOCALTIMESTAMP` is executed to observe the differences in format.

Note: 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.

Comparing Date and Time in a Session's Time Zone

Results of queries:

session SET altered.

SESSIONTIMEZONE	CURRENT_DATE
1 -05:00	10-SEP-2012 23:08:16

1

SESSIONTIMEZONE	CURRENT_TIMESTAMP
1 -05:00	10-SEP-12 11.10.21.183775000 PM -05:00

2

SESSIONTIMEZONE	LOCALTIMESTAMP
1 -05:00	10-SEP-12 11.10.43.871626000 PM

3

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In this case, the `CURRENT_DATE` function returns the current date in the session's time zone, the `CURRENT_TIMESTAMP` function returns the current date and time in the session's time zone as a value of the data type `TIMESTAMP WITH TIME ZONE`, and the `LOCALTIMESTAMP` function returns the current date and time in the session's time zone.

Note: The code example output may vary depending on when the command is run.

DBTIMEZONE and SESSIONTIMEZONE

- Display the value of the database time zone:

```
SELECT DBTIMEZONE FROM DUAL;
```

DBTIMEZONE
1 +00:00

- Display the value of the session's time zone:

```
SELECT SESSIONTIMEZONE FROM DUAL;
```

SESSIONTIMEZONE
1 -05:00

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The DBA sets 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 cannot 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 in the slide shows that the database time zone is set to "-05:00," 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 offset to UTC by -5 hours. Observe that the database time zone is different from the current session's time zone.

TIMESTAMP Data Types

Data Type	Fields
TIMESTAMP	Year, Month, Day, Hour, Minute, Second with fractional seconds
TIMESTAMP WITH TIME ZONE	Same as the TIMESTAMP data type; also includes: TIMEZONE_HOUR, and TIMEZONE_MINUTE or TIMEZONE_REGION
TIMESTAMP WITH LOCAL TIME ZONE	Same as the TIMESTAMP data type; also includes a time zone offset in its value

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The **TIMESTAMP** data type is an extension of the **DATE** data type.

TIMESTAMP (fractional_seconds_precision)

This data type contains the year, month, and day values of date, as well as hour, minute, and second values of time, where significant fractional seconds precision is the number of digits in the fractional part of the **SECOND** datetime field. The accepted values of significant **fractional_seconds_precision** are 0 through 9. The default is 6.

TIMESTAMP (fractional_seconds_precision) WITH TIME ZONE

This data type contains all values of **TIMESTAMP** as well as time zone displacement value.

TIMESTAMP (fractional_seconds_precision) WITH LOCAL TIME ZONE

This data type contains all values of **TIMESTAMP**, 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.

TIMESTAMP Fields

Datetime Field	Valid Values
YEAR	–4712 to 9999 (excluding year 0)
MONTH	01 to 12
DAY	01 to 31
HOUR	00 to 23
MINUTE	00 to 59
SECOND	00 to 59.9(N) where 9(N) is precision
TIMEZONE_HOUR	–12 to 14
TIMEZONE_MINUTE	00 to 59

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Each datetime data type is composed of several of these fields. Datetimes are mutually comparable and assignable only if they have the same datetime fields.

Difference Between DATE and TIMESTAMP

A

```
-- when hire_date is  
of type DATE  
  
SELECT hire_date  
FROM emp4;
```

	HIRE_DATE
1	17-JUN-03
2	21-SEP-05
3	13-JAN-01
4	03-JAN-06
5	21-MAY-07
6	25-JUN-05
7	05-FEB-06
8	07-FEB-07
9	17-AUG-02
10	16-AUG-02

...

B

```
ALTER TABLE emp4  
MODIFY hire_date  
TIMESTAMP(7);  
SELECT hire_date  
FROM emp4;
```

	HIRE_DATE
1	17-JUN-03 12.00.00.000000000 AM
2	21-SEP-05 12.00.00.000000000 AM
3	13-JAN-01 12.00.00.000000000 AM
4	03-JAN-06 12.00.00.000000000 AM
5	21-MAY-07 12.00.00.000000000 AM
6	25-JUN-05 12.00.00.000000000 AM
7	05-FEB-06 12.00.00.000000000 AM
8	07-FEB-07 12.00.00.000000000 AM
9	17-AUG-02 12.00.00.000000000 AM
10	16-AUG-02 12.00.00.000000000 AM

...

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

TIMESTAMP Data Type: Example

In the slide, example A shows the data from the `hire_date` column of the `EMP4` table when the data type of the column is `DATE`. In example B, the table is altered and the data type of the `hire_date` column is made into `TIMESTAMP`. The output shows the differences in display. You can convert from `DATE` to `TIMESTAMP` when the column has data, but you cannot convert from `DATE` or `TIMESTAMP` to `TIMESTAMP WITH TIME ZONE` unless the column is empty.

You can specify the fractional seconds precision for time stamp. If none is specified, as in this example, it defaults to 6.

For example, the following statement sets the fractional seconds precision as 7:

```
ALTER TABLE emp4  
MODIFY hire_date TIMESTAMP(7);
```

Note: The Oracle `DATE` data type by default looks like what is shown in this example. However, the date data type also contains additional information such as hours, minutes, seconds, AM, and PM. To obtain the date in this format, you can apply a format mask or a function to the date value.

Comparing TIMESTAMP Data Types

```
CREATE TABLE web_orders  
(order_date TIMESTAMP WITH TIME ZONE,  
delivery_time TIMESTAMP WITH LOCAL TIME ZONE);
```

```
INSERT INTO web_orders values  
(current_date, current_timestamp + 2);
```

```
SELECT * FROM web_orders;
```

ORDER_DATE	DELIVERY_TIME
1 10-SEP-12 11.30.20.000000000 PM -05:00	12-SEP-12 11.30.20.000000000 PM

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, a new table `web_orders` is created with a column of data type `TIMESTAMP WITH TIME ZONE` and a column of data type `TIMESTAMP WITH LOCAL TIME ZONE`. This table is populated whenever a `web_order` is placed. The time stamp and time zone for the user placing the order are inserted based on the `CURRENT_DATE` value. The local time stamp and time zone are populated by inserting two days from the `CURRENT_TIMESTAMP` value into it every time an order is placed. When a web-based company guarantees shipping, it can estimate its delivery time based on the time zone of the person placing the order.

Note: The code example output may vary as per the time of run of the command.

Lesson Agenda

- CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP
- INTERVAL data types
- Using the following functions:
 - EXTRACT
 - TZ_OFFSET
 - FROM_TZ
 - TO_TIMESTAMP
 - TO_YMINTERVAL
 - TO_DSINTERVAL

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

INTERVAL Data Types

- INTERVAL data types are used to store the difference between two datetime values.
- There are two classes of intervals:
 - Year-month
 - Day-time
- The precision of the interval is:
 - The actual subset of fields that constitutes an interval
 - Specified in the interval qualifier

Data Type	Fields
INTERVAL YEAR TO MONTH	Year, Month
INTERVAL DAY TO SECOND	Days, Hour, Minute, Second with fractional seconds

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

INTERVAL data types are used to store the difference between two datetime values. There are two classes of intervals: year-month intervals and day-time intervals. A year-month interval is made up of a contiguous subset of fields of `YEAR` and `MONTH`, whereas a day-time interval is made up of a contiguous subset of fields consisting of `DAY`, `HOURL`, `MINUTE`, and `SECOND`. The actual subset of fields that constitute an interval is called the precision of the interval and is specified in the interval qualifier. Because the number of days in a year is calendar-dependent, the year-month interval is NLS-dependent, whereas day-time interval is NLS-independent.

The interval qualifier may also specify the leading field precision, which is the number of digits in the leading or only field, and in case the trailing field is `SECOND`, it may also specify the fractional seconds precision, which is the number of digits in the fractional part of the `SECOND` value. If not specified, the default value for leading field precision is 2 digits, and the default value for fractional seconds precision is 6 digits.

INTERVAL YEAR (year_precision) TO MONTH

This data type stores a period of time in years and months, where `year_precision` is the number of digits in the `YEAR` datetime field. The accepted values are 0 through 9. The default is 6.

INTERVAL DAY (day_precision) TO SECOND (fractional_seconds_precision)

This data type stores a period of time in days, hours, minutes, and seconds, where `day_precision` is the maximum number of digits in the `DAY` datetime field (accepted values are 0 through 9; the default is 2), and `fractional_seconds_precision` is the number of digits in the fractional part of the `SECOND` field. The accepted values are 0 through 9. The default is 6.

INTERVAL Fields

INTERVAL Field	Valid Values for Interval
YEAR	Any positive or negative integer
MONTH	00 to 11
DAY	Any positive or negative integer
HOURL	00 to 23
MINUTE	00 to 59
SECOND	00 to 59.9(N) where 9(N) is precision

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

INTERVAL YEAR TO MONTH can have fields of YEAR and MONTH.

INTERVAL DAY TO SECOND can have fields of DAY, HOUR, MINUTE, and SECOND.

The actual subset of fields that constitute an item of either type of interval is defined by an interval qualifier, and this subset is known as the precision of the item.

Year-month intervals are mutually comparable and assignable only with other year-month intervals, and day-time intervals are mutually comparable and assignable only with other day-time intervals.

INTERVAL YEAR TO MONTH: Example

```
CREATE TABLE warranty
(prod_id number, warranty_time INTERVAL YEAR(3) TO
MONTH);

INSERT INTO warranty VALUES (123, INTERVAL '8' MONTH);
INSERT INTO warranty VALUES (155, INTERVAL '200'
YEAR(3));
INSERT INTO warranty VALUES (678, '200-11');
SELECT * FROM warranty;
```

	PROD_ID	WARRANTY_TIME
1	123	0-8
2	155	200-0
3	678	200-11

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

INTERVAL YEAR TO MONTH stores a period of time using the YEAR and MONTH datetime fields. Specify INTERVAL YEAR TO MONTH 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.

Restriction: The leading field must be more significant than the trailing field. For example, INTERVAL '0-1' MONTH TO YEAR is not valid.

Examples

- INTERVAL '123-2' YEAR(3) TO MONTH
Indicates an interval of 123 years, 2 months
- INTERVAL '123' YEAR(3)
Indicates an interval of 123 years, 0 months
- INTERVAL '300' MONTH(3)
Indicates an interval of 300 months
- INTERVAL '123' YEAR
Returns an error because the default precision is 2, and '123' has 3 18

The Oracle database supports two interval data types: `INTERVAL YEAR TO MONTH` and `INTERVAL DAY TO SECOND`; the column type, PL/SQL argument, variable, and return type must be one of the two. However, for interval literals, the system recognizes other American National Standards Institute (ANSI) interval types such as `INTERVAL '2' YEAR` or `INTERVAL '10' HOUR`. In these cases, each interval is converted to one of the two supported types.

In the example in the slide, a `WARRANTY` table is created, which contains a `warranty_time` column that takes the `INTERVAL YEAR (3) TO MONTH` data type. Different values are inserted into it to indicate years and months for various products. When these rows are retrieved from the table, you see a year value separated from the month value by a (-).

INTERVAL DAY TO SECOND Data Type: Example

```
CREATE TABLE lab  
( exp_id number, test_time INTERVAL DAY(2) TO SECOND);  
  
INSERT INTO lab VALUES (100012, '90 00:00:00');  
INSERT INTO lab VALUES (56098,  
    INTERVAL '6 03:30:16' DAY TO SECOND);
```

```
SELECT * FROM lab;
```

	EXP_ID	TEST_TIME
1	100012	90 0:0:0.0
2	56098	6 3:30:16.0

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

In the example in the slide, you create the `lab` table with a `test_time` column of the `INTERVAL DAY TO SECOND` data type. You then insert into it the value `'90 00:00:00'` to indicate 90 days and 0 hours, 0 minutes, and 0 seconds, and `INTERVAL '6 03:30:16' DAY TO SECOND` to indicate 6 days, 3 hours, 30 minutes, and 16 seconds. The `SELECT` statement shows how this data is displayed in the database.

Lesson Agenda

- CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP
- INTERVAL data types
- Using the following functions:
 - EXTRACT
 - TZ_OFFSET
 - FROM_TZ
 - TO_TIMESTAMP
 - TO_YMINTERVAL
 - TO_DSINTERVAL

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

EXTRACT

- Display all employees who were hired after 2007.

```
SELECT last_name, employee_id, hire_date
FROM employees
WHERE EXTRACT(YEAR FROM TO_DATE(hire_date, 'DD-MON-RR')) > 2007
ORDER BY hire_date;
```

- 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;
```

	1 LAST_NAME	2 HIRE_DATE	3 EXTRACT(MONTH FROM HIRE_DATE)
1	Kochhar	21-SEP-05	9
2	De Haan	13-JAN-01	1
3	Raphaely	07-DEC-02	12
4	Weiss	18-JUL-04	7
5	Fripp	10-APR-05	4
6	Kauffman	01-MAY-03	5

...

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 { expr } )
```

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 a date in the Gregorian calendar. When extracting from a datetime with a time zone value, the value returned is in UTC.

In the first example in the slide, the EXTRACT function is used to select all employees who were hired after 2007. In the second example in the slide, the EXTRACT function is used to extract the MONTH from the HIRE_DATE column of the EMPLOYEES table for those employees who report to the manager whose EMPLOYEE_ID is 100.

TZ_OFFSET

Display the time zone offset for the 'US/Eastern',
'Canada/Yukon' and 'Europe/London' time zones:

```
SELECT TZ_OFFSET('US/Eastern'),  
       TZ_OFFSET('Canada/Yukon'),  
       TZ_OFFSET('Europe/London')  
FROM DUAL;
```

	TZ_OFFSET('US/EASTERN')	TZ_OFFSET('CANADA/YUKON')	TZ_OFFSET('EUROPE/LONDON')
1	-04:00	-07:00	+01:00

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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, this value indicates that the time zone where the command was executed is eight hours behind 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 : mi'  
          | SESSIONTIMEZONE  
          | DBTIMEZONE  
          })
```

The Fold Motor Company has its headquarters in Michigan, USA, which is in the US/Eastern time zone. The company president, Mr. Fold, wants to conduct a conference call with the vice president of the Canadian operations and the vice president of European operations, who are in the Canada/Yukon and Europe/London time zones, respectively. Mr. Fold wants to find out the time in each of these places to make sure that his senior management will be available to attend the meeting. His secretary, Mr. Scott, helps by issuing the queries shown in the example and gets the following results:

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

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

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

	 TZNAME	 TZABBREV	 CON_ID
1	Africa/Abidjan	LMT	0
2	Africa/Abidjan	GMT	0
3	Africa/Accra	LMT	0
4	Africa/Accra	GMT	0
5	Africa/Accra	GHST	0

...

FROM_TZ

Display the **TIMESTAMP** value '2000-07-12 08:00:00' as a **TIMESTAMP WITH TIME ZONE** value for the 'Australia/North' time zone region.

```
SELECT FROM_TZ(TIMESTAMP
                '2000-07-12 08:00:00', 'Australia/North')
FROM DUAL;
```

FROM_TZ(TIMESTAMP'2000-07-12 08:00:00','AUSTRALIA/NORTH')
1 12-JUL-00 08.00.00.000000000 AM AUSTRALIA/NORTH

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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_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 an optional TZD format. TZD is an abbreviated time zone string with daylight saving information. TZR represents the time zone region in datetime input strings. Examples are 'Australia/North', 'PST' for US/Pacific standard time, 'PDT' for US/Pacific daylight time, and so on.

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

Note: To see a listing of valid values for the TZR and TZD format elements, query the **V\$TIMEZONE_NAMES** dynamic performance view.

TO_TIMESTAMP

Display the character string '2007-03-06 11:00:00' as a `TIMESTAMP` value:

```
SELECT TO_TIMESTAMP ('2007-03-06 11:00:00',
                     'YYYY-MM-DD HH:MI:SS')
FROM DUAL;
```

TO_TIMESTAMP('2007-03-06 11:00:00','YYYY-MM-DD HH:MI:SS')
1 06-MAR-07 11.00.00.000000000 AM

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

The `TO_TIMESTAMP` function converts a string of `CHAR`, `VARCHAR2`, `NCHAR`, or `NVARCHAR2` data type to a value of the `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 in the slide converts a character string to a value of `TIMESTAMP`.

Note: You use the `TO_TIMESTAMP_TZ` function to convert a string of `CHAR`, `VARCHAR2`, `NCHAR`, or `NVARCHAR2` data type to a value of the `TIMESTAMP WITH TIME ZONE` data type. For more information about this function, see *Oracle Database SQL Language Reference* for Oracle Database 12c.

TO_YMINTERVAL

Display a date that is one year and 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_YMININTERVAL
1	17-FEB-04	17-APR-05
2	17-AUG-05	17-OCT-06

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 and two months after the hire date for the employees working in the department 20 of the EMPLOYEES table.

TO_DSINTERVAL

Display a date that is 100 days and 10 hours after the hire date for all the employees.

```
SELECT last_name,  
       TO_CHAR(hire_date, 'mm-dd-yy:hh:mi:ss') hire_date,  
       TO_CHAR(hire_date +  
               TO_DSINTERVAL('100 10:00:00'),  
               'mm-dd-yy:hh:mi:ss') hiredate2  
FROM employees;
```

	LAST_NAME	HIRE_DATE	HIREDATE2
1	King	06-17-03:12:00:00	09-25-03:10:00:00
2	Kochhar	09-21-05:12:00:00	12-30-05:10:00:00
3	De Haan	01-13-01:12:00:00	04-23-01:10:00:00
4	Hunold	01-03-06:12:00:00	04-13-06:10:00:00
5	Ernst	05-21-07:12:00:00	08-29-07:10:00:00
6	Austin	06-25-05:12:00:00	10-03-05:10:00:00
7	Pataballa	02-05-06:12:00:00	05-16-06:10:00:00
8	Lorentz	02-07-07:12:00:00	05-18-07:10:00:00
9	Greenberg	08-17-02:12:00:00	11-25-02:10:00:00
10	Faviet	08-16-02:12:00:00	11-24-02:10:00:00
11	Chen	09-28-05:12:00:00	01-06-06:10:00:00

...

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

TO_DSINTERVAL converts a character string of the CHAR, VARCHAR2, NCHAR, or NVARCHAR2 data type to an INTERVAL DAY TO SECOND data type.

In the example in the slide, the date 100 days and 10 hours after the hire date is obtained.

Daylight Saving Time (DST)

- Start of Daylight Saving:
 - Time jumps from 01:59:59 AM to 03:00:00 AM.
 - Values from 02:00:00 AM to 02:59:59 AM are not valid.
- End of Daylight Saving:
 - Time jumps from 02:00:00 AM to 01:00:01 AM.
 - Values from 01:00:01 AM to 02:00:00 AM are ambiguous because they are visited twice.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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 start of Daylight Saving to the end of Daylight Saving in 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 same time.

The Oracle database automatically determines, for any given time zone region, whether daylight saving time is in effect and returns local time values accordingly. The datetime value is sufficient for the Oracle database to determine whether daylight saving time is in effect for a given region in all cases except boundary cases. A boundary case occurs during the period when daylight saving time goes into or out of effect. For example, in the US/Eastern region, when daylight saving time goes into effect, the time changes from 01:59:59 AM to 03:00:00 AM. The one-hour interval between 02:00:00 AM and 02:59:59 AM. does not exist. When daylight saving time goes out of effect, the time changes from 02:00:00 AM back to 01:00:01 AM, and the one-hour interval between 01:00:01 AM and 02:00:00 AM is repeated.

ERROR_ON_OVERLAP_TIME

The `ERROR_ON_OVERLAP_TIME` is a session parameter to notify the system to issue an error when it encounters a datetime that occurs in the overlapped period and no time zone abbreviation was specified to distinguish the period.

For example, daylight saving time ends on October 31, at 02:00:01 AM. The overlapped periods are:

- 10/31/2004 01:00:01 AM to 10/31/2004 02:00:00 AM (EDT)
- 10/31/2004 01:00:01 AM to 10/31/2004 02:00:00 AM (EST)

If you input a datetime string that occurs in one of these two periods, you need to specify the time zone abbreviation (for example, EDT or EST) in the input string for the system to determine the period. Without this time zone abbreviation, the system does the following:

If the `ERROR_ON_OVERLAP_TIME` parameter is `FALSE`, it assumes that the input time is standard time (for example, EST). Otherwise, an error is raised.

Quiz

The `TIME_ZONE` session parameter may be set to:

- a. A relative offset
- b. Database time zone
- c. OS local time zone
- d. A named region

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

Answer: b, c, d

Summary

In this lesson, you should have learned how to:

- Use data types similar to `DATE` that store fractional seconds and track time zones
- Use data types that store the difference between two datetime values
- Use the following datetime functions:
 - `CURRENT_DATE`
 - `CURRENT_TIMESTAMP`
 - `LOCALTIMESTAMP`
 - `DBTIMEZONE`
 - `SESSIONTIMEZONE`
 - `EXTRACT`
 - `TZ_OFFSET`
 - `FROM_TZ`
 - `TO_TIMESTAMP`
 - `TO_YMINTERVAL`
 - `TO_DSINTERVAL`

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

This lesson addressed some of the datetime functions available in the Oracle database.

Practice 10: Overview

This practice covers using the datetime functions.

ORACLE

Copyright © 2014, Oracle and/or its affiliates. All rights reserved.

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

YASINEB MAZOUZ (z.m.malak@hotmail.com) has a non-transferable
license to use this Student Guide.