

ORACLE SQL

Merge statements, Time zones

Lesson Objectives

- To understand the following topics:

- Merge statement
- 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 datetime functions:



2.1: Merge

MERGE statement

- The MERGE statement, provides the ability to conditionally update or insert data into a database table.
- The MERGE statement, performs an UPDATE if the row exists, and an INSERT if it is a new row:
 - Increases performance and ease of use
 - Is useful in data warehousing applications
 - Avoids separate updates



Copyright © Capgemini 2015. All Rights Reserved 3

MERGE Statement

The MERGE statement is used mostly in data warehouse environments to build the data in warehouse

2.1: Merge

MERGE statement

- You can conditionally insert or update rows in a table by using the MERGE statement

```
MERGE INTO table_name table_alias
USING (table|view|sub_query) alias
ON (join condition) WHEN MATCHED THEN
UPDATE SET
  col1 = col_val1,
  col2 = col2_val
WHEN NOT MATCHED THEN
INSERT (column_list)
VALUES (column_values);
```



Copyright © Capgemini 2015. All Rights Reserved 4

INTO : this is how we specify the target for the MERGE. The target must be either a table or an updateable view (an in-line view cannot be used here);

USING : the USING clause represents the source dataset for the MERGE. This can be a single table (as in our example) or an in-line view;

ON () : the ON clause is where we supply the join between the source dataset and target table. Note that the join conditions must be in parentheses;

WHEN MATCHED: this clause is where we instruct Oracle on what to do when we already have a matching record in the target table (i.e. there is a join between the source and target datasets). We obviously want an UPDATE in this case. One of the restrictions of this clause is that we cannot update any of the columns used in the ON clause (though of course we don't need to as they already match). Any attempt to include a join column will raise an unintuitive invalid identifier exception; and

WHEN NOT MATCHED : this clause is where we INSERT records for which there is no current match.

Note that sqlplus reports the number of rows merged. This includes both the updates and inserts. Oracle treats MERGE as a MERGE and not an UPDATE+INSERT statement. The same is true of SQL%ROWCOUNT in PL/SQL.

2.1: Merge

Example on Merge

■ Example

```
CREATE table staff_copy as select staff_code,staff_name FROM  
staff_master where 1=2;
```

```
MERGE into staff_copy using staff_master  
ON (staff_master.deptno=staff_copy.deptno)  
WHEN MATCHED THEN  
UPDATE SET staff_code=staff_master.staff_code,  
staff_name=staff_master.staff_name  
WHEN NOT MATCHED THEN  
INSERT (staff_code,staff_name) values  
(staff_master.staff_code,staff_master.staff_name);
```

2.2: Time Zones

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';
```



Copyright © Capgemini 2015. All Rights Reserved 6

TIME_ZONE Session Parameter

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.

2.2: Time Zones

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



Copyright © Capgemini 2015. All Rights Reserved 7

CURRENT_DATE, CURRENT_TIMESTAMP, and LOCALTIMESTAMP

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 11g: SQL Fundamentals I.

2.2: Time Zones

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;  
SELECT SESSIONTIMEZONE, CURRENT_TIMESTAMP FROM DUAL;  
SELECT SESSIONTIMEZONE, LOCALTIMESTAMP FROM DUAL;
```

1

2

3



Copyright © Capgemini 2015. All Rights Reserved 8

Comparing Date and Time in a Session's Time Zone

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.

2.2: Time Zones

Comparing Date and Time in a Session's Time Zone

Results of queries:

```
ALTER SESSION succeeded.
```

	SESSIONTIMEZONE	CURRENT_DATE
1	-05:00	23-JUN-2009 01:34:52

1

	SESSIONTIMEZONE	CURRENT_TIMESTAMP
1	-05:00	23-JUN-09 01.35.26.239882000 AM -05:00

2

	SESSIONTIMEZONE	LOCALTIMESTAMP
1	-05:00	23-JUN-09 01.36.21.811798000 AM

3



Copyright © Capgemini 2015. All Rights Reserved 9

Comparing Date and Time in a Session's Time Zone (continued)

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.

2.2: Time Zones

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

 Copyright © Capgemini 2015. All Rights Reserved. 10

DBTIMEZONE and SESSIONTIMEZONE

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 -8 hours. Observe that the database time zone is different from the current session's time zone.

2.2: Time Zones

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

Capgemini
CONSULTING TECHNOLOGY OBSERVATORY

Copyright © Capgemini 2015. All Rights Reserved 11

TIMESTAMP Data Types

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.

2.2: Time Zones

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

TIMESTAMP Fields

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.

2.2: Time Zones

Difference Between DATE and TIMESTAMP

A

```
-- when hire_date is of type DATE

SELECT hire_date
FROM employees;
```

	HIRE_DATE
1	21-JUN-99
2	13-JAN-00
3	17-SEP-87
4	17-FEB-96
5	17-AUG-97
6	07-JUN-94
7	07-JUN-94
8	07-JUN-94

B

```
ALTER TABLE employees
MODIFY hire_date TIMESTAMP;

SELECT hire_date
FROM employees;
```

	HIRE_DATE
1	21-JUN-99 12.00.00.000000000 AM
2	13-JAN-00 12.00.00.000000000 AM
3	17-SEP-87 12.00.00.000000000 AM
4	17-FEB-96 12.00.00.000000000 AM
5	17-AUG-97 12.00.00.000000000 AM
6	07-JUN-94 12.00.00.000000000 AM
7	07-JUN-94 12.00.00.000000000 AM
8	07-JUN-94 12.00.00.000000000 AM

Capgemini
CONSULTING TECHNOLOGY SERVICES

Copyright © Capgemini 2015. All Rights Reserved 13

TIMESTAMP Data Type: Example

In the slide, example A shows the data from the hire_date column of the EMPLOYEES 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 employees
MODIFY hire_date TIMESTAMP(7);
```

Note: The Oracle date data type by default appears as 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.

2.2: Time Zones

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	23-JUN-09 01.56.39.000000000 AM -05:00	25-JUN-09 01.56.39.000000000 AM



Copyright © Capgemini 2015. All Rights Reserved 14

Comparing TIMESTAMP Data Types

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 is inserted based on the `CURRENT_DATE` value. The local time stamp and time zone is 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, they can estimate their delivery time based on the time zone of the person placing the order.

2.2: Time Zones

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



Copyright © Capgemini 2015. All Rights Reserved 15

INTERVAL Data Types

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, HOUR, 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.

2.2: Time Zones

INTERVAL Fields

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



Copyright © Capgemini 2015. All Rights Reserved 16

INTERVAL Fields

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.

2.2: Time Zones

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



Copyright © Capgemini 2015. All Rights Reserved 17

INTERVAL YEAR TO MONTH Data Type

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 three digits

2.2: Time Zones

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



Copyright © Capgemini 2015. All Rights Reserved 18

INTERVAL DAY TO SECOND Data Type: Example

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.

2.2: Time Zones

EXTRACT

- Display the YEAR component from the SYSDATE.

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

EXTRACT(YEARFROMSYSDATE)
2009

- 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)
1	Hartstein	17-FEB-1996 00:00:00	2
2	Kochhar	21-SEP-1989 00:00:00	9
3	De Haan	13-JAN-1993 00:00:00	1
4	Raphaely	07-DEC-1994 00:00:00	12
5	Weiss	18-JUL-1996 00:00:00	7



Copyright © Capgemini 2015. All Rights Reserved 19

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 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 extract the YEAR from SYSDATE. 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.

2.2: Time Zones

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



Copyright © Capgemini 2015. All Rights Reserved. 20

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, 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:mm' ]
           [ 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.

2.2: Time Zones

FROM_TZ

- Display the TIMESTAMP value '2000-03-28 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



Copyright © Capgemini 2015. All Rights Reserved 21

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 'TZR: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.

2.2: Time Zones

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;
```

 Capgemini
CONSULTING TECHNOLOGY SERVICES

Copyright © Capgemini 2015. All Rights Reserved 22

TO_TIMESTAMP

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 11g Release 1 (11.1).

2.2: Time Zones

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;
```



Copyright © Capgemini 2015. All Rights Reserved 23

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

2.2: Time Zones

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	OConnell	06-21-99:12:00:00	09-29-99:10:00:00
2	Grant	01-13-00:12:00:00	04-22-00:10:00:00
3	Whalen	09-17-87:12:00:00	12-26-87:10:00:00
4	Hartstein	02-17-96:12:00:00	05-27-96:10:00:00
5	Fay	08-17-97:12:00:00	11-25-97:10:00:00
6	Mavris	06-07-94:12:00:00	09-15-94:10:00:00
7	Baer	06-07-94:12:00:00	09-15-94:10:00:00
8	Higgins	06-07-94:12:00:00	09-15-94:10:00:00



Copyright © Capgemini 2015. All Rights Reserved 24

TO_DSINTERVAL

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.

2.2: Time Zones

Daylight Saving Time

- First Sunday in April
 - 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.
- Last Sunday in October
 - 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.



Copyright © Capgemini 2015. All Rights Reserved 25

Daylight Saving Time (DST)

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 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.

Summary

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

- CURRENT_DATE
- CURRENT_TIMESTAMP
- LOCALTIMESTAMP
- DBTIMEZONE
- SESSIONTIMEZONE
- EXTRACT
- TZ_OFFSET
- FROM_TZ
- TO_TIMESTAMP
- TO_YMINTERVAL
- TO_DSINTERVAL



Copyright © Capgemini 2015. All Rights Reserved 26

Summary

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

Review Question

- Question 1: _____ data types are used to store the difference between two datetime values.
- Question 2: _____ statement, provides the ability to conditionally update or insert data into a database table.



Add the notes here.