

Time Zones

From the Big Picture

To the Small Picture

By

Edward Kosciuszko

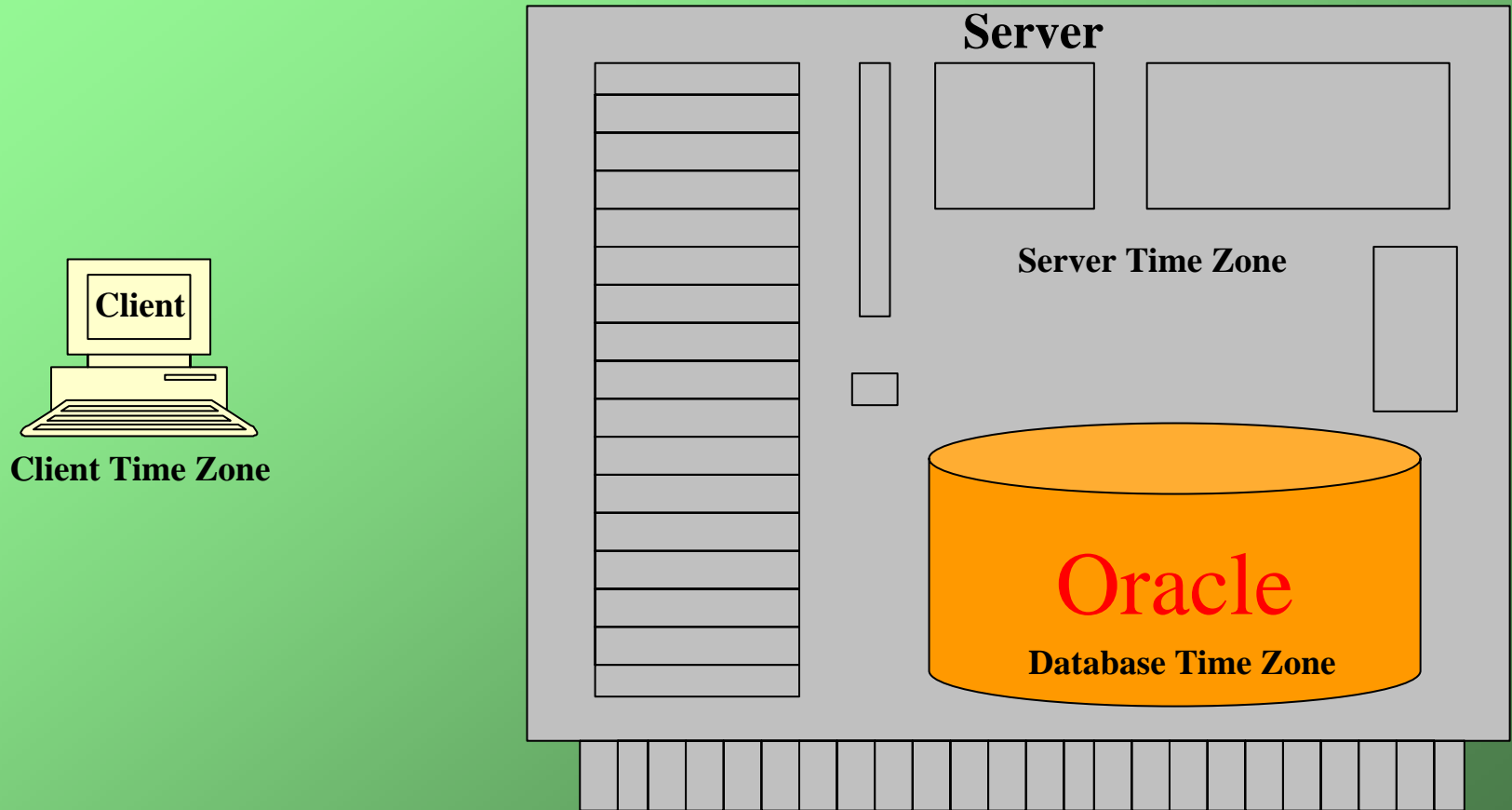
Kosware Inc.

12/9/2007

Sequel@Optonline.net

TIME ZONES

3 Different Time Zones



TIME ZONES

Considerations

- **Date literal or current time**
- **Time is relative to user**
- **Current time in EST is different from PST**
- **5:00 AM EST and 2:00 AM PST are equal**
- **2007-11-04 01:30AM – is this the time before or after daylight savings took affect?**

TIME ZONES

And More Considerations

- **Date indicates when entered relative to user?**
- **What to do when time zone algorithm changes?**
- **Replication?**

TIME ZONES

Database Time Zone

CREATE DATABASE ... SET TIMEZONE='GMT'

- **Default is OS time zone offset (which means database time zone setting can be different if created during DST or not.)**
- **Used to normalize `TIMESTAMP WITH LOCAL TIME ZONE` and `TIMESTAMP WITH TIME ZONE` values**

RECOMMENDATION

Always set to explicit offset (e.g. `-4:00`)

TIME ZONES

Database Time Zone

```
ALTER DATABASE SET TIMEZONE='EST'
```

- **Only possible if no TIMESTAMP WITH LOCAL TIME ZONE columns**
- **Requires DB restart to take affect.**
- **DBTIMEZONE function references database time zone**

TIME ZONES

Session Time Zone

ALTER SESSION SET TIME_ZONE = local

- sets to O/S time zone

ALTER SESSION SET TIME_ZONE = DBTIMEZONE

- sets to database time zone

ALTER SESSION SET TIME_ZONE = '-05:00'

- sets to explicit offset from GMT

ALTER SESSION SET TIME_ZONE = 'America/New_York'

- sets to named region

TIME ZONES

4 Different Date Types

- **DATE**
- **TIMESTAMP**
- **TIMESTAMP WITH TIME ZONE**
- **TIMESTAMP WITH TIME ZONE**

TIME ZONES

DATE

- Year to second but no time zone
- Great for historical dates or dates based on server only.
- 7 bytes

TIMESTAMP

- DATE w/ fractional seconds; no time zone
- argument = fractional digits (default=6 max=9)
- default format: `NLS_TIMESTAMP_FORMAT`
- 7 to 11 bytes storage

TIME ZONES

TIMESTAMP WITH TIME ZONE

- **timestamp w/ time zone offset**
- **timestamp is normalized to UTC**
- **default format: NLS_TIMESTAMP_TZ_FORMAT**
- **cannot be part of primary key or unique index
(Oracle considers 2 dates that map to same UTC
time to be equal.)**
- **fixed 13 bytes**

TIME ZONES

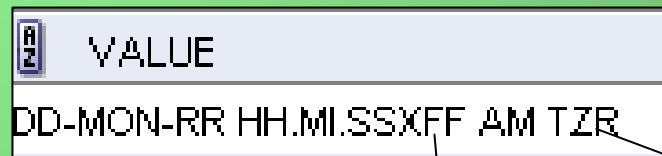
TIMESTAMP WITH LOCAL TIME ZONE

- **time normalized to DBTIMEZONE**
- **originating time zone is not stored**
- **default format: NLS_TIMESTAMP_TZ_FORMAT**
- **7 - 11 bytes depending on fractional seconds**

TIME ZONES

TIMESTAMP WITH TIME ZONE

```
SELECT value FROM v$nls_parameters  
WHERE parameter = 'NLS_TIMESTAMP_TZ_FORMAT'
```



A screenshot of a database query result. The first row is highlighted in light blue and contains the text 'VALUE'. The second row contains the format string 'DD-MON-RR HH.MI.SSXXFF AM TZR'. Two lines with arrows point from the 'SSXXFF' and 'TZR' parts of the format string to red boxes labeled 'Fractional seconds' and 'Time zone region' respectively.

Entering literal:

```
INSERT .....VALUES (... , '12-MAR-06 10:00:00AM',...)
```

Default time zone: session time zone

TIME ZONES

TIMESTAMP WITH TIME ZONE

Literal Defaults

```
CREATE TABLE timestamp_w_tz
(sess_tz VARCHAR2(30), tz_time TIMESTAMP WITH TIME ZONE);
ALTER SESSION SET TIME_ZONE = 'America/New_York'
INSERT INTO timestamp_w_tz values (SESSIONTIMEZONE,
                                   '12-MAR-06 10:00:00AM')
ALTER SESSION SET TIME_ZONE = 'US/Pacific'
INSERT INTO timestamp_w_tz values (SESSIONTIMEZONE,
                                   '12-MAR-06 10:00:00AM')
SELECT SESSIONTIMEZONE, sess_tz, tz_time FROM timestamp_w_tz
```

Session time
zone

No time zone
entered

SESSIONTIMEZONE	SESS_TZ	TZ_TIME
US/Pacific	America/New_York	12-MAR-06 10.00.00.000000000 AM AMERICA/NEW_YORK
US/Pacific	US/Pacific	12-MAR-06 10.00.00.000000000 AM US/PACIFIC

TIME ZONES

TIMESTAMP WITH TIME ZONE

Default Display = Entered Date

```
ALTER SESSION SET TIME_ZONE = 'America/New_York'
```

```
INSERT INTO timestamp_w_tz values (SESSIONTIMEZONE,  
    TO_TIMESTAMP_TZ ('20060312 10:00:00 America/New_York',  
        'YYYYMMDD HH24:MI:SS TZR'))
```

```
ALTER SESSION SET TIME_ZONE = 'US/Pacific'
```

```
INSERT INTO timestamp_w_tz values (SESSIONTIMEZONE,  
    TO_TIMESTAMP_TZ ('20060312 10:00:00 America/New_York',  
        'YYYYMMDD HH24:MI:SS TZR'))
```

```
SELECT SESSIONTIMEZONE, sess_tz, tz_time FROM timestamp_w_tz
```

SESSIONTIMEZONE	SESS_TZ	TZ_TIME
US/Pacific	US/Pacific	12-MAR-06 10.00.00.000000000 AM AMERICA/NEW_YORK
US/Pacific	America/New_York	12-MAR-06 10.00.00.000000000 AM AMERICA/NEW_YORK

TIME ZONES

TIMESTAMP WITH TIME ZONE INTERNAL STORAGE FORMAT

```
INSERT INTO timestamp_w_tz values ('AMERICA/NEW_YORK',  
    TO_TIMESTAMP_TZ ('20060312 10:00:00 America/New_York',  
                    'YYYYMMDD HH24:MI:SS TZR')));  
  
INSERT INTO timestamp_w_tz values ('US/PACIFIC',  
    TO_TIMESTAMP_TZ ('20060312 10:00:00 US/Pacific',  
                    'YYYYMMDD HH24:MI:SS TZR'))  
  
INSERT INTO timestamp_w_tz values ('CANADA/EASTERN',  
    TO_TIMESTAMP_TZ ('20060312 10:00:00 Canada/Eastern',  
                    'YYYYMMDD HH24:MI:SS TZR'))  
  
INSERT INTO timestamp_w_tz values ('-5:00',  
    TO_TIMESTAMP_TZ ('20060312 10:00:00 -5:00',  
                    'YYYYMMDD HH24:MI:SS TZH:TZM'))  
  
INSERT INTO timestamp_w_tz values ('-4:00',  
    TO_TIMESTAMP_TZ ('20060312 11:00:00 -4:00',  
                    'YYYYMMDD HH24:MI:SS TZH:TZM'))
```

Time zone by
explicit offset

TIME ZONES

TIMESTAMP WITH TIME ZONE INTERNAL STORAGE FORMAT

'12-MAR-06 11.00.00.000000000 AM +00:00'

is stored as

120,106,3,12,12,1,1,0,0,0,0,20,60

1st byte = century (excess 100 notation); e.g. 120-100 = 20

2nd byte = year in century (excess 100 notation) ; e.g. 106-100 = 06

3rd byte = month; e.g. 3 = March

4th byte = day; e.g. 12

5th byte = hour (excess 1 notation); e.g. 12-1 = 11

6th byte = minute (excess 1 notation); e.g. 1-1 = 0

7th byte = second (excess 1 notation); e.g. 1-1 = 0

8th – 11th for fractional seconds – recall 2 digits per byte

12th & 13th for time zone; e.g. 20,60 indicates +00:00

TIME ZONES

TIMESTAMP WITH TIME ZONE INTERNAL STORAGE FORMAT

- All timestamps normalized relative to UTC (GMT)
- America/New_York and Canada/Eastern have same offset but ID stored differently in 12th & 13th byte.
- Each value is unique but cannot create unique index

16 - 1 = 15
which is the
GMT hour

SESS_TZ	TZ_TIME	DUMP(TZ_TIME)
AMERICA/NEW_YORK	12-MAR-06 10.00.00.000000000 AM AMERICA/NEW_YORK	Typ=181 Len=13: 120,106,3,12,16,1,1,0,0,0,0,129,144
US/PACIFIC	12-MAR-06 10.00.00.000000000 AM US/PACIFIC	Typ=181 Len=13: 120,106,3,12,19,1,1,0,0,0,0,137,156
CANADA/EASTERN	12-MAR-06 10.00.00.000000000 AM CANADA/EASTERN	Typ=181 Len=13: 120,106,3,12,16,1,1,0,0,0,0,137,232
-5:00	12-MAR-06 10.00.00.000000000 AM -05:00	Typ=181 Len=13: 120,106,3,12,16,1,1,0,0,0,0,15,60
-4:00	12-MAR-06 11.00.00.000000000 AM -04:00	Typ=181 Len=13: 120,106,3,12,16,1,1,0,0,0,0,16,60
-0:00	12-MAR-06 11.00.00.000000000 AM +00:00	Typ=181 Len=13: 120,106,3,12,12,1,1,0,0,0,0,20,60

TIME ZONES

TIMESTAMP WITH LOCAL TIME ZONE Default Display Relative to Session

```
CREATE TABLE timestamp_w_ltz
(sess_tz VARCHAR2(30), tz_time TIMESTAMP WITH LOCAL TIME ZONE);

ALTER SESSION SET TIME_ZONE = 'America/New_York'

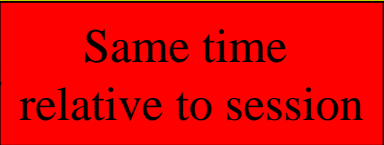
INSERT INTO timestamp_w_ltz values (SESSIONTIMEZONE,
                                   '12-MAR-06 10:00:00AM')

ALTER SESSION SET TIME_ZONE = 'US/Pacific'

INSERT INTO timestamp_w_ltz values (SESSIONTIMEZONE,
                                   '12-MAR-06 10:00:00AM')

ALTER SESSION SET TIME_ZONE = 'GMT'

INSERT INTO timestamp_w_ltz values (SESSIONTIMEZONE,
                                   '12-MAR-06 10:00:00AM')
```



Same time
relative to session

TIME ZONES

TIMESTAMP WITH LOCAL TIME ZONE Default Display Relative to Session

```
ALTER SESSION SET TIME_ZONE = 'America/New_York'
```

```
SELECT SESSIONTIMEZONE, sess_tz, tz_time FROM timestamp_w_tz
```

SESSIONTIMEZONE	SESS_TZ	TZ_TIME
America/New_York	US/Pacific	12-MAR-06 01.00.00.000000000 PM
America/New_York	GMT	12-MAR-06 05.00.00.000000000 AM
America/New_York	America/New_York	12-MAR-06 10.00.00.000000000 AM

```
ALTER SESSION SET TIME_ZONE = 'US/Pacific'
```

```
SELECT SESSIONTIMEZONE, sess_tz, tz_time FROM timestamp_w_tz
```

SESSIONTIMEZONE	SESS_TZ	TZ_TIME
US/Pacific	US/Pacific	12-MAR-06 10.00.00.000000000 AM
US/Pacific	GMT	12-MAR-06 02.00.00.000000000 AM
US/Pacific	America/New_York	12-MAR-06 07.00.00.000000000 AM

TIME ZONES

TIMESTAMP WITH LOCAL TIME ZONE INTERNAL STORAGE FORMAT

- All dates stored in DBTIMEZONE
- Originating time zone not stored

```
ALTER SESSION SET TIME_ZONE = 'America/New_York'  
  
SELECT DBTIMEZONE, sess_tz, TZ_OFFSET(sess_tz),  
       tz_time, dump(tz_time)  
FROM timestamp_w_ltz
```

11-1=10

DBTIMEZONE	SESS_TZ	TZ_OFFSET(SESS_TZ)	TZ_TIME	DUMP(TZ_TIME)
-05:00	America/New_York	-05:00	12-MAR-06 10.00.00.000000000 AM	Typ=231 Len=7: 120,106,3,12,11,1,1
-05:00	US/Pacific	-08:00	12-MAR-06 01.00.00.000000000 PM	Typ=231 Len=7: 120,106,3,12,14,1,1
-05:00	GMT	+00:00	12-MAR-06 05.00.00.000000000 AM	Typ=231 Len=7: 120,106,3,12,6,1,1

TIME ZONES

Session Time

- **SYSDATE** is only the date taken directly from the server
- **SYSTIMESTAMP** is **SYSDATE** with time zone (return type is **TIMESTAMP WITH TIME ZONE**)

```
SELECT sysdate, dbtimezone, systimestamp  
FROM dual
```

TO_CHAR(SYSDATE,'DD-MON-YY...)	DBTIMEZONE	SYSTIMESTAMP
26-nov-2007 09:38:06 PM	-05:00	26-NOV-07 09.38.06.046000000 PM -04:00

NOTE server time zone is -4:00

TIME ZONES

Session Time

- **LOCALTIMESTAMP** is current date and time relative to session time zone
- Cast as **TIMESTAMP** (i.e. time zone is lost) !!!

```
ALTER SESSION SET TIME_ZONE = '-3:00'
```

```
SELECT dbtimezone, systimestamp, localtimestamp  
FROM dual
```

DBTIMEZONE	SYSTIMESTAMP	LOCALTIMESTAMP
-05:00	27-NOV-07 06.59.22.859000000 AM -05:00	27-NOV-07 08.59.22.859000000 AM

NOTE server time zone is **-5:00**

TIME ZONES

Session Time

```
INSERT INTO timestamp_w_tz
VALUES (sessiontimezone, localtimestamp)
SELECT * FROM timestamp_w_tz
```

SESS_TZ	TZ_TIME
-03:00	27-NOV-07 09.05.44.703000000 AM -03:00

```
INSERT INTO timestamp_w_ltz
VALUES (sessiontimezone, localtimestamp)
SELECT * FROM timestamp_w_tz
```

SESS_TZ	TZ_TIME
-03:00	27-NOV-07 09.09.21.765000000 AM

Time zone picked up from session

TIME ZONES

Time Zone Conversion

```
SELECT sess_tz, tz_time, tz_time AT TIME ZONE '+5:00'  
FROM timestamp_w_tz
```

SESS_TZ	TZ_TIME	TZ_TIME AT TIMEZONE '+5:00'
-03:00	27-NOV-07 09.05.44.703000000 AM -03:00	27-NOV-07 05.05.44.703000000 PM +05:00

Session Time
Zone

```
ALTER SESSION SET TIME_ZONE = '-4:00'
```

```
SELECT sess_tz, tz_time, tz_time AT LOCAL  
FROM timestamp_w_tz
```

SESS_TZ	DBTIMEZONE	TZ_TIME	TZ_TIME AT LOCAL
-3:00	-05:00	27-NOV-07 09.05.44.703000000 AM -03:00	27-NOV-07 08.05.44.703000000 AM -04:00

TIME ZONES

Time Zone Conversion

```
SELECT sess_tz, tz_time, tz_time
       AT TIME ZONE 'US/Pacific' FROM timestamp_w_tz
```

SESS_TZ	TZ_TIME	TZ_TIME AT TIME ZONE 'US/PACIFIC'
-03:00	28-NOV-07 08.58.14.593000000 AM -03:00	28-NOV-07 03.58.14.593000000 AM US/PACIFIC

... AT TIME ZONE DBTIMEZONE ...

... AT TIME ZONE SESSIONTIMEZONE ...

TIME ZONES

Time Zone Conversion

```
SELECT sess_tz, tz_time,  
CAST(tz_time AS TIMESTAMP WITH LOCAL TIME ZONE)  
FROM timestamp_w_tz
```

SESS_TZ	TZ_TIME	CAST(TZ_TIME AS TIMESTAMP WITH LOCAL TIME ZONE)
-03:00	28-NOV-07 08.58.14.593000000 AM -03:00	28-NOV-07 08.58.14.593000000 AM

CAST(...AS TIMESTAMP)

CAST(... AS TIMESTAMP WITH TIME ZONE)

CAST(... AS TIMESTAMP WITH LOCAL TIME ZONE)

TIME ZONES

Time Zone Conversion

Standard \longleftrightarrow Daylight Savings

```
ALTER SESSION SET NLS_DATE_FORMAT =  
                'DD-MON-YYYY HH24:MI:SS'  
  
SELECT NEW_TIME('29-OCT-2006 01:23:45', 'EDT', 'EST')  
FROM DUAL;
```

NEW_DATE
29-OCT-2006 00:23:45

TIME ZONES

CURRENT_TIMESTAMP

CURRENT_DATE

- **current datetime in session time zone but no time zone**

CURRENT_TIMESTAMP

- **CURRENT_DATE plus session time zone**
- **use instead of SYSDATE when recording client datetime in applications**

TIME ZONES

Criteria Performance

```
CREATE TABLE sales_copy
(prod_id NUMBER(6,0), cust_id NUMBER,
time_tz TIMESTAMP WITH TIME ZONE,
time_ltz TIMESTAMP WITH LOCAL TIME ZONE,
channel_id CHAR(1 BYTE), promo_id NUMBER(6,0),
quantity_sold NUMBER(3,0), amount_sold NUMBER(10,2) )

ALTER SESSION SET TIME_ZONE = 'America/New_York'

INSERT INTO sales_copy
SELECT prod_id, cust_id,
       to_timestamp(to_char(time_id,'DD-MON-RR HH:MI:SS AM')) at local,
       to_timestamp(to_char(time_id,'DD-MON-RR HH:MI:SS AM')) at local,
... FROM sales
```

- Based on Oracle's SH demo account table, SALES.
- Tests on 9i (-5:00, 16K blocks) – 10g (-00:00) 8K

Criteria Performance

When executing
INSERT in
SQL*Developer.

ORA-01878: specified field not found in datetime or interval

?

Checking trace of login to SQL*Plus vs SQL*Developer:

- SQL*Plus sets TIME_ZONE = '-5:00'
- SQL*Developer sets TIME_ZONE = 'America/New_York'
- April 5 2:00:00AM – April 5 2:59:59AM does NOT exist

```
SELECT time_tz FROM sales_copy WHERE time_tz  
BETWEEN to_timestamp_tz('05-APR-98 2:00:00 AM -5:00')  
AND to_timestamp_tz('05-APR-98 2:59:59 AM -5:00')
```

TIME_TZ
05-APR-98 02.00.00.0000000000 AM -05:00
05-APR-98 02.00.00.0000000000 AM -05:00

Bad Data!

Criteria Performance

Problem is that original data contained only EST times

How do you convert only the rows involving the DST change?

- 1. Convert each date time to GMT**
- 2. Convert step 1 to `TIMESTAMP` string**
- 3. Append `'+00:00'` to string and convert to `TIMESTAMP WITH TIME ZONE`**
- 4. Add `"AT LOCAL"` to convert `TIMESTAMP` in step 3 to the local time zone, `America/New_York`**

Criteria Performance

```
SELECT to_timestamp_tz (  
  to_char(new_time(  
    to_date('02-apr-2006 2:59:59AM','DD-MON-YYYY HH:MI:SS AM'),  
    'EST','GMT'),  
    'YYYY-MM-DD HH:MI:SS AM')||' +00:00',  
  'YYYY-MM-DD HH:MI:SSAM TZH:TZM') AT LOCAL
```

FROM dual

```
TO_TIMESTAMP_TZ(TO_CHAR(NEW_TIME(TO_DATE('02-APR-06 03.59.59.000000000 AM AMERICA/NEW_YORK
```

Hour added for DST

```
SELECT to_timestamp_tz (  
  to_char(new_time(  
    to_date('02-apr-2006 1:59:59AM','DD-MON-YYYY HH:MI:SS AM'),  
    'EST','GMT'),  
    'YYYY-MM-DD HH:MI:SS AM')||' +00:00',  
  'YYYY-MM-DD HH:MI:SSAM TZH:TZM') AT LOCAL
```

FROM dual

```
TO_TIMESTAMP_TZ(TO_CHAR(NEW_TIME(TO_DATE('02-APR-06 01.59.59.000000000 AM AMERICA/NEW_YORK
```

Hour NOT added
because EST

Generic DATE to TIME ZONE Conversion

```
INSERT INTO sales_copy  
SELECT prod_id, cust_id,  
to_timestamp_tz (  
to_char(new_time(time_id,'EST','GMT'), 'YYYY-MM-DD HH:MI:SS AM')||' +00:00',  
'YYYY-MM-DD HH:MI:SSAM TZH:TZM') AT LOCAL,  
to_timestamp_tz (  
to_char(new_time(time_id,'EST','GMT'), 'YYYY-MM-DD HH:MI:SS AM')||' +00:00',  
'YYYY-MM-DD HH:MI:SSAM TZH:TZM') AT LOCAL, ...  
FROM sales_no_part
```

- **SQL Assumes ALL date times are EST**
- **If only portion are EST, use WHERE criteria to identify**
- **NEW_TIME returns DATE**

TIME ZONES

Criteria Performance

NO Indexes

9i
Results

```
ALTER SESSION SET TIME_ZONE = 'US/Pacific'
```

```
SELECT count(*) FROM sales_copy  
WHERE time_tz > '06-JAN-98 12:00:00 AM'
```

SQL_TEXT	EXECUTIONS	CPU_TIME	ELAPSED_TIME
SELECT count(*) FROM sales_copy WHERE time_tz > '0...	1	515625	539552

```
ALTER SESSION SET TIME_ZONE = 'US/Pacific'
```

```
SELECT count(*) FROM sales_copy  
WHERE time_ltz > '06-JAN-98 12:00:00 AM'
```

SQL_TEXT	EXECUTIONS	CPU_TIME	ELAPSED_TIME
SELECT count(*) FROM sales_copy WHERE time_ltz > '06...	1	250000	246115

Significant performance gain

52% less
CPU

Continue >

TIME ZONES

Criteria Performance

NO Indexes

Convert constant to same time zone as that stored?

ALTER SESSION SET TIME_ZONE = 'US/Pacific'

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
copy WHERE time_tz > from_tz (TIMESTAMP '1998-01-06 00:00:00', 'GMT')	1	515625	509568
copy WHERE time_tz > from_tz(timestamp '1998-01-06 00:00:00', 'US/Pacific')	1	593750	583497
copy WHERE time_tz > from_tz (TIMESTAMP '1998-01-06 00:00:00', 'America/New_York')	1	421875	421973
copy WHERE time_tz > '06-JAN-98 12:00:00 AM'	1	515625	518099

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
where time_tz > from_tz(timestamp '1998-01-06 00:00:00','GMT')	1	241497	244511
where time_tz > '06-JAN-98 12:00:00 AM'	1	336833	336960
where time_tz > from_tz(timestamp '1998-01-06 00:00:00','America/New_York')	1	238975	238975
where time_tz > from_tz(timestamp '1998-01-06 00:00:00','US/Pacific')	1	237224	237224

10g – Explicitly convert to time zone – 29% gain

TIME ZONES

Criteria Performance

NO Indexes

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
WHERE time_tz > '06-JAN-98 12:00:00 AM'	1	265625	257240
WHERE time_tz > from_tz (TIMESTAMP '1998-01-06 00:00:00', 'GMT')	1	312500	299739
WHERE time_tz > from_tz (TIMESTAMP '1998-01-06 00:00:00', 'America/New_York')	1	234375	229383
WHERE time_tz > from_tz(timestamp '1998-01-06 00:00:00', 'US/Pacific')	1	312500	317272

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
where time_tz > from_tz(timestamp '1998-01-06 00:00:00','GMT')	1	6467805	6471821
where time_tz > from_tz(timestamp '1998-01-06 00:00:00','US/Pacific')	1	6469163	6489110
where time_tz > '06-JAN-98 12:00:00 AM'	1	139098	167518
where time_tz > from_tz(timestamp '1998-01-06 00:00:00','America/New_York')	1	6462113	6465838

10g: DATE is 64 % faster than TIMESTAMP WITH LOCAL TIME ZONE!

**10g
bug?**

TIME ZONES

Indexed Criteria Performance

	EXECUTI...	CPU_TIME	ELAPSED_TIME
where time_id > '30-DEC-2001'	1	9607	9607
where time_tz > '30-DEC-2001 12:00:00AM'	1	1342	1342
where time_ttz > '30-DEC-2001 12:00:00AM'	1	1313	1313

786 rows

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
where time_tz > '01-NOV-2001 12:00:00AM'	1	10818	10818
where time_ttz > '01-NOV-2001 12:00:00AM'	1	8729	8729
not where time_id > '01-NOV-2001'	2	25520	25520

44,070 rows

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
where time_tz > '06-JAN-1998 12:00:00AM'	1	157379	165408
where time_ttz > '06-JAN-1998 12:00:00AM'	1	171972	171972
part where time_id > '06-JAN-1998'	1	135648	142393

917,177 rows

TIMESTAMP index performs better when fewer rows.

TIME ZONES

TIMESTAMP WITH TIME ZONE - INDEX

```
CREATE INDEX i_tz ON sales_copy (time_tz)
```

- generates the following function-based index:

```
CREATE INDEX "SH"."I_TZ" ON "SH"."SALES_COPY"  
(SYS_EXTRACT_UTC("TIME_TZ"))
```

- **SYS_EXTRACT_UTC** converts to UTC timestamp
- Query does not require **SYS_EXTRACT_UTC** to gain access to index.

TIME ZONES

Timestamp Arithmetic

```
WHERE current_timestamp - time_ltz > 100
```

ORA-00932: inconsistent datatypes: expected INTERVAL got NUMBER

```
WHERE current_timestamp - time_ltz >  
      INTERVAL '100 0:0:0' DAY(3) TO SECOND
```

- Difference of 2 timestamps is INTERVAL
- DAY(3) required since default digits for days is 2

TIME ZONES

Timestamp Arithmetic

NO Indexes

WHERE current_timestamp - time_ttz >
INTERVAL '100 0:0:0' DAY(3) TO SECOND

WHERE current_timestamp - INTERVAL '100 0:0:0' DAY(3) TO SECOND
> time_ttz

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
WHERE current_timestamp - time_ttz > INTERVAL '100 0:0:0' DAY(3) TO SECOND	9i 1	23093750	23092805
WHERE current_timestamp - INTERVAL '100 0:0:0' DAY(3) TO SECOND > time_ttz	1	33562500	33646460

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
WHERE current_timestamp - INTERVAL '100 0:0:0' DAY(3) TO SECOND > time_ttz	10g 1	22094661	22229865
WHERE current_timestamp - time_ttz > INTERVAL '100 0:0:0' DAY(3) TO SECOND	1	21747998	21816859

- **9i** results make no sense
- “current_timestamp – INTERVAL” makes more sense since performed only once for all rows evaluated

TIME ZONES

Timestamp Arithmetic

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
where current_timestamp > time_tz > INTERVAL '100 0:0:0' DAY(3) TO SECOND	1	16681053	16694403
where current_timestamp - INTERVAL '100 0:0:0' DAY(3) TO SECOND > time_tz	1	17137657	17191052
where current_timestamp > time_tz > INTERVAL '100 0:0:0' DAY(3) TO SECOND	1	11360730	11383211
where current_timestamp - INTERVAL '100 0:0:0' DAY(3) TO SECOND > time_tz	1	8402531	8428084

- 10g test

NO Indexes

- **TIMESTAMP WITH TIME ZONE > 2x's faster**

**WHERE CAST(current_timestamp - INTERVAL '100 0:0:0' DAY(3)
TO SECOND AS TIMESTAMP WITH LOCAL TIME ZONE)
> time_tz**

2.25 x's slower !?!?

TIME ZONES

Timestamp Arithmetic

Indexes
Used

```
WHERE CAST(current_timestamp - INTERVAL '100 0:0:0' DAY(3)
          TO SECOND AS TIMESTAMP WITH LOCAL TIME ZONE)
> time_tz
```

	EXECUTIONS	CPU_TIME	ELAPSED_TIME
AY(3) TO SECOND AS TIMESTAMP WITH LOCAL TIME ZONE) > time_tz	1	220664	222474
AY(3) TO SECOND AS TIMESTAMP WITH LOCAL TIME ZONE) > time_tz	1	251748	288555
e + 100 > time_id	1	371418	371546

With indexes **TIMESTAMP WITH TIME ZONE**
is **40%** faster than **DATE**

TIME ZONES

Daylight Savings Time

2006 Daylight Savings Time Schedule

Start: (+1 hour) April 2, 2006 2:00:00 AM

End: (-1 hour) October 29, 2006 2:00:00 AM

On Apr 2nd, 2:00:00 – 2:59:59 AM does not exist

On Oct 29th, 1:00:00 – 1:59:59 AM is repeated

Boundary Problem

How do you distinguish the repeated times on Oct 29th?

TIME ZONES

Daylight Savings Time

Apr 2nd, 2:00:00 – 2:59:59 AM does not exist

```
SELECT to_char(to_date('02-APR-2006 2:00:01 AM',
  'DD-MON-YYYY HH:MI:SS AM'), 'DD-MON-YYYY HH:MI:SS AM')
FROM dual
```

```
TO_CHAR(TO_DATE('02-APR-2006 2:00:01 AM',
  'DD-MON-YYYY HH:MI:SS AM'), 'DD-MON-YYYY HH:MI:SS AM')
02-APR-2006 02:00:01 AM
```

Time doesn't exist!

```
SELECT to_timestamp_tz('02-APR-2006 1:59:59 AM America/New_York',
  'DD-MON-YYYY HH:MI:SS AM TZR')
FROM dual
```

```
TO_TIMESTAMP_TZ('02-APR-2006 1:59:59 AM AMERICA/NEW_YORK',
  'DD-MON-YYYY HH:MI:SS AM TZR')
02-APR-06 01.59.59.000000000 AM AMERICA/NEW_YORK
```

TIME ZONES

Daylight Savings Time

Apr 2nd, 2:00:00 – 2:59:59 AM does not exist

```
SELECT to_timestamp_tz('02-APR-2006 1:59:59 AM America/New_York',  
'DD-MON-YYYY HH:MI:SS AM TZR') + INTERVAL '0 0:0:1' DAY TO SECOND  
FROM dual
```

```
TO_TIMESTAMP_TZ('02-APR-20061:59:59AMAMERICA/NEW  
02-APR-06 03.00.00.000000000 AM AMERICA/NEW_YORK')
```

Advances hour
due to DST

```
SELECT to_timestamp_tz('02-APR-2006 2:00:00 AM America/New_York',  
'DD-MON-YYYY HH:MI:SS AM TZR') FROM dual
```

```
ORA-01878: specified field not found in datetime or interval
```

TIME ZONES

Daylight Savings Time

Oct 29th, 1:00:00 – 1:59:59 AM is repeated

```
SELECT to_timestamp_tz('29-OCT-2006 1:30:00 AM America/New_York', 'DD-MON-YYYY HH:MI:SS AM TZR TZD') FROM dual
```

```
ORA-01883: overlap was disabled during a region transition
```

```
ALTER SESSION SET ERROR_ON_OVERLAP_TIME = FALSE
```

Now “overlap” times are considered to be standard time (in this case EST).

Default: FALSE
Recommendation: Set TRUE to prohibit implicit entry.

TIME ZONES

Daylight Savings Time

Oct 29th, 1:00:00 – 1:59:59 AM is repeated

```
SELECT to_char(to_timestamp_tz('29-OCT-2006 1:30:00 AM America/New_York',
                               'DD-MON-YYYY HH:MI:SS AM TZR'),
           'DD-MON-YYYY HH:MI:SS AM TZR TZD')
FROM dual
```

```
TO_CHAR(TO_TIMESTAMP_TZ('29-OCT-2006 1:30:00 AM AMERICA/NEW_YORK EST',
                        'DD-MON-YYYY HH:MI:SS AM TZR TZD'))
```

TZD requests the DST value to be printed

How do you indicate time is EDT?

TIME ZONES

Daylight Savings Time

Forcing DST

```
SELECT to_timestamp_tz('29-OCT-2006 1:30:00 AM America/New_York EDT',  
  'DD-MON-YYYY HH:MI:SS AM TZR TZD') FROM dual
```

```
TO_TIMESTAMP_TZ('29-OCT-20061:30:00AMAMERICA/NEW  
29-OCT-06 01.30.00.000000000 AM AMERICA/NEW_YORK
```

DST specified

```
SELECT to_timestamp_tz('29-OCT-2006 2:30:00 AM America/New_York EDT',  
  'DD-MON-YYYY HH:MI:SS AM TZR TZD') FROM dual
```

```
ORA-01857: not a valid time zone
```

DST ended at 1:59:59 EDT, so this is not a valid “EDT” time.

TIME ZONES

Daylight Savings Time

- DST information is limited in future and may change
- Explicitly specify DST for future dates

```
SELECT * FROM v$timezone_names WHERE tzname = 'America/New_York'
```

TZNAME	TZABBREV
America/New_York	LMT
America/New_York	EST
America/New_York	EWT
America/New_York	EDT

TIME ZONES

Daylight Savings Time

- **Environmental variable, ORA_TZFILE, identifies Oracle time zone file.**
- **2 files: in \$ORACLE_HOME/oracore/zoneinfo**
- **timezonelrg.dat contains all time zones**
- **timezone.dat contains common time zones**
- **Some clients contain time zone files (not thin JDBC client)**
- **Oracle JVM has time zone information embedded**
- **Java Runtime Environment (JRE) stores rules on DST**
- **Upgrade both ORACLE and OJVM for 2007**

TIME ZONES

Default Behavior

**DATE data
type**

```
SELECT to_timestamp(time_id),  
       to_timestamp(to_char(time_id,'DD-MON-RR HH:MI:SS AM'))  
FROM sales
```

TO_TIMESTAMP(TIME_ID)	TO_TIMESTAMP(TO_CHAR(TIME_ID,'DD-MON-RR HH:MI:SS AM'))
28-JAN-98 12.00.00.000000000 AM	28-JAN-98 02.00.00.000000000 PM
28-JAN-98 12.00.00.000000000 AM	28-JAN-98 08.00.00.000000000 AM
28-JAN-98 12.00.00.000000000 AM	28-JAN-98 07.00.00.000000000 AM
28-JAN-98 12.00.00.000000000 AM	28-JAN-98 12.00.00.000000000 PM
28-JAN-98 12.00.00.000000000 AM	28-JAN-98 10.00.00.000000000 PM
28-JAN-98 12.00.00.000000000 AM	28-JAN-98 03.00.00.000000000 PM

**W/o TO_CHAR,
date truncated
before converting
to timestamp**

9.2.0.7.0

Name	Value
NUM_ROWS	1016271
BLOCKS	3400
AVG_ROW_LEN	47

10.2.0.1.0

Name	Value
NUM_ROWS	918843
BLOCKS	6228
AVG_ROW_LEN	47

TIME ZONES

Default Behavior

```
SELECT systimestamp, systimestamp + 1 FROM dual
```

SYSTIMESTAMP	SYSTIMESTAMP+1
30-NOV-07 06.20.49.578000000 PM -05:00	01-DEC-07

Not the default
TIMESTAMP
format

```
SELECT TO_CHAR(systimestamp + 1,  
              'DD-MON-RR HH:MI:SS TZR')  
FROM dual
```

```
ORA-01821: date format not recognized
```

Addition changed **TIMESTAMP** to **DATE** data type.

```
SELECT systimestamp + INTERVAL '1 0:0:0' DAY TO SECOND FROM dual
```

SYSTIMESTAMP+INTERVAL'1 0:0:0'DAYTOSECOND
01-DEC-07 06.19.27.953000000 PM -05:00

TIME ZONES

Which Datatype?

TIMESTAMP WITH TIME ZONE

- **can determine when the data was entered relative to client**
- **date arithmetic more efficient since already in UTC**
- **must force transformation to session time zone**

TIMESTAMP WITH LOCAL TIME ZONE

- **automatically adjusts for client time zone querying**
- **replication: DB's must have same DBTIMEZONE**
- **date arithmetic forces conversion to UTC**

TIME ZONES

Questions ?

12/9/2007

Edward Kosciuszko
SeQueL@Optonline.net