

ATTENZIONE: Anche se ho impiegato cura ed impegno nella realizzazione di questo documento, consiglio vivamente chiunque voglia applicare i suggerimenti da me proposti, di eseguire tutte le opportune prove in un ambiente di test, prima di effettuare eventuali modifiche in produzione. Questa convinzione nasce dal fatto che nonostante mi sia documentato attraverso libri di testo, manuali e documenti in rete, esiste sempre la probabilità di eventuali errori nella stesura ed imprecisioni nella esposizione.

ATTENZIONE: Quanto scritto in questo documento può essere utilizzato in modo del tutto libero. Pregherei comunque tutti coloro che desiderano delle modificarlo, di comunicarlo al [mio](mailto:mio) indirizzo di posta in modo tale da permettermi di aggiornare e/o correggere eventuali errori.

<http://www.oral.com> Aprile 2005

Di [Andrea Salzano](#)

## Time Zones

Le ore del giorno sono calcolate in base alla rotazione della terra. Il tempo del giorno in un particolare momento, dipende da dove sei. La terra è divisa in 24 zone temporali, time zones da qui in avanti, una per ogni ora del giorno. Il tempo lungo il primo meridiano a Greenwich (Inghilterra), è noto come Greenwich mean time, o GMT. GMT è riferito anche come UTC, Universal Time Coordinated.

GMT è il tempo standard a cui tutti i time zones della terra fanno riferimento. E' lo stesso durante tutto l'anno e non è influenzato dall'ora solare (daylitgh saving time) e l'ora legale (summer time).

Il meridiano è una linea immaginaria che inizia dal polo nord e termina al polo sud. Il primo meridiano, che come detto passa per Greenwich, è anche noto come longitudine zero ed è la linea da cui tutte le altre longitudini sono misurate. Il tempo è quindi misurato relativamente a GMT e tutti i luoghi sono identificati da una latitudine (la loro distanza dal nord o dal sud dell'equatore) e da una longitudine (la distanza da est od ovest dal meridiano di Greenwich). Informazioni utili relativamente ai time zones, sono reperibili al seguente link: <http://wgp.greenwichmeantime.com/home.htm>.

### 1. Ora Solare

Molte nazioni occidentali, avanzano l'orologio di un'ora durante i mesi estivi. Questo periodo è definito come "ora solare" (in inglese, daylitgh saving time). Negli Stati Uniti, nel Messico ed in Canda, il daylitgh saving time, inizia la prima domenica di Aprile e termina l'ultima domenica di Ottobre. In Europa invece, l'ora Solare inizia una settimana prima rispetto al daylitgh saving time, mentre termina lo stesso giorno.

### 2. Supporto per il Time Zone

Per far fronte ai Time Zone, Oracle 9i ha introduce nuovi tipi di dato da aggiungere alla DATA:

- `TIMESTAMP`, che supporta le frazioni di secondo
- `TIMESTAMP WITH TIME ZONE (TSTZ)`, che supporta le frazioni di secondo ed inoltre mostra il valore delle ore e dei minuti prima o dopo UTC
- `TIMESTAMP WITH LOCAL TIME ZONE (TSLTZ)`, che è come TSTZ ma con le seguenti eccezioni:
  - I dati sono normalizzati nel *database time zone*, quando sono memorizzati nel database
  - Quando i dati sono selezionati, l'utente vede i dati aggiustati nel *session time zone*

Come si è potuto notare, sono stati introdotti due nuovi concetti: il `DATABASE TIME ZONE` ed il `SESSION TIME ZONE`.

Il primo indica il time zone del db e per default è lo stesso del sistema operativo. E' impostato specificando la clausola `SET TIME_ZONE` nello statement `CREATE DATABASE`. Se omesso, viene impostato a quello del sistema operativo. Il secondo indica il time zone della sessione corrente ovvero del client con cui ci si connette al db.

Per identificare i valori del time zone del database e della sessione, esistono due nuove funzioni: `DBTIMEZONE` e `SESSIONTIMEZONE`. Tali funzioni possono essere utilizzati sulla tabella `DUAL`:

```
select DBTIMEZONE, SESSIONTIMEZONE from dual;
```

Il tipo restituito dalla funzione `DBTIMEZONE` è un offset, un insieme di caratteri nel formato `'[+|-] TZH:TZM'`, o il time zone espresso in un nome di regione (questo dipende da come l'utente ha specificato il database time zone quando ha creato il db). Anche per la funzione `SESSIONTIMEZONE`, il tipo restituito è un offset nella forma `'[+|-] TZH:TZM'`, o il time zone espresso in un nome di regione. In questo caso però la visualizzazione dipende da come il client ha specificato il time zone nello statement `ALTER SESSION`.

Cerchiamo di approfondire il concetto di `OFFSET`. Come intuitivamente si può capire, l'`OFFSET` è il tempo di scarto tra il tempo locale di una regione rispetto al meridiano di Greenwich. Roma ad esempio ha come `OFFSET` `" +01:00 "` (`+02:00`, durante l'ora solare). Oracle automaticamente determina se l'ora solare è attiva per una data regione, e restituisce il "tempo" locale in modo corretto. In linea di massima all'`RDBMS` è sufficiente l'orario (inteso come tempo completo di anno, mese, giorno, ore, minuti, secondi) per determinare se una data regione è sotto ora solare o meno. Questo è vero sempre eccetto nei casi in cui c'è il passaggio tra ora solare ed ora legale e viceversa. Nel primo caso infatti, il tempo passa da 2:00 a.m. a 3:00 a.m. : l'ora di intervallo tra le 2:00 a.m. e le 3:00 a.m. non esiste. Nel

secondo caso invece, l'ora si sposta all'indietro dalle 2:00 a.m. all' 1:00 a.m. : qui l'ora di intervallo tra 1:00 a.m. e 2:00 a.m. è ripetuto.

Come si è potuto capire da quanto fin qui detto, il TIME\_ZONE, può essere espresso in funzione della regione o dell'OFFSET. Abbiamo già detto che l'OFFSET è espresso nella forma '[+ | -] TZH:TZM' (per Roma, TZH=01, mentre TZM=00). Le regioni invece sono espresse come 'regione timezone'. La lista di tutte le regioni è possibile ottenerla interrogando (anche come utente non DBA) la vista V\$TIMEZONE\_NAMES. Per Roma, la regione è espressa come Europe/Rome<sup>1</sup>.

Esiste una funzione, la TZ\_OFFSET, che restituisce l'OFFSET del corrispondente valore inserito. La sintassi per tale funzione è infatti:

```
TZ_OFFSET
( { 'time_zone_name'
  | '{ + | - } hh : mi'
  | SESSIONTIMEZONE
  | DBTIMEZONE
  }
)
```

Posso quindi conoscere l'OFFSET per una particolare regione oppure quello relativo al time zone del database o della sessione.

Come esempio consideriamo un db installato a Roma ed eseguiamo lo statement il 2 Aprile (con in vigore quindi l'ora solare):

```
SQL> select tz_offset('Europe/Rome') from dual;
TZ_OFFS
-----
+02:00
```

```
SQL> select dbtimezone,sessiontimezone,sysdate from dual;
DBTIME SESSIONTIMEZONE      SYSDATE
-----
+02:00 +02:00                01-04-05 16.07.44
```

---

<sup>1</sup> Di default Oracle utilizza un insieme ristretto di regioni. Queste sono definite all'interno del file binario, \$ORACLE\_HOME/oracore/zoneinfo/timezone.dat. Ci si può rendere facilmente conto di ciò, eseguendo una count sulla V\$TIMEZONE\_NAMES. Se il risultato restituito è di 616, allora si sta utilizzando l'insieme ristretto di regioni. Per usare quello esteso occorre, come indicato nella nota 227334.1 di Metalink, impostare la variabile d'ambiente ORA\_TSFIL=\$ORACLE\_HOME/oracore/zoneinfo/timezlr.dat e far ripartire il db. A quanto sembra tale file viene letto una sola volta allo startup. Per rendersi conto a questo punto dell'uso esteso delle regioni da parte di Oracle, la count sulla \$TIMEZONE\_NAMES deve restituire 1250. Se si utilizza l'insieme ristretto delle regioni, la zona 'Europe/Rome' non esiste.

Da cui si evince che l'OFFSET per Roma è di +2 ore rispetto a UTC (durante l'ora legale è di una sola ora). Il database inoltre è installato con un TIMEZONE di +02:00 che coincide con il TIMEZONE del client che ha eseguito lo statement. Il giorno (inteso nella sua forma completa anno, mese, giorno, ore, minuti, secondi) è indicato nella colonna SYSDATE<sup>2</sup>.

Faccio notare che:

1) select DBTIMEZONE from dual

e

2) select TZ\_OFFSET (DBTIMEZONE) from dual

sono equivalenti, così come

3) select SESSIONTIMEZONE from dual

e

4) select TZ\_OFFSET (SESSIONTIMEZONE) from dual

Il motivo risiede nel fatto che la funzione TZ\_OFFSET accetta come valori: un nome di regione, un offset di una regione (in questo caso restituisce se stesso) o le parole chiavi DBTIMEZONE e SESSIONTIMEZONE. La 1) restituisce un time zone nella forma '[+ | -] TZH:TZM', che passato a TZ\_OFFSET, restituisce proprio DBTIMEZONE. Lo stesso è vero per SESSIONTIMEZONE.

Esistono altre tre funzioni che aiutano a determinare il "tempo" nel nostro database. Queste sono: CURRENT\_DATE, CURRENT\_TIMESTAMP, LOCALTIMESTAMP. Prima di proseguire occorre fare una premessa. Prima di Oracle 9, il tempo era memorizzato come tipo DATE, storicizzato internamente in 7 bytes. In particolare, l'informazione era mantenuta come: secolo, anno, mese, giorno, ora, minuto, secondo. La seguente tabella può dare un'ulteriore chiarificazione:

BYTE	Meaning	
----	-----	
1	Century	-- stored in excess-100 notation
2	Year	-- " "
3	Month	-- stored in 0 base notation
4	Day	-- " "
5	Hour	-- stored in excess-1 notation
6	Minute	-- " "
7	Second	-- " "

Il tipo di dato TIMESTAMP invece è stato introdotto per consentire di utilizzare

---

<sup>2</sup> Per prelevare la data, "sysdate" esegue una "system call" verso il Sistema Operativo (una chiamata "gettimeofday"). Questo vuol dire che "sysdate" non usa i timezones del db (DBTIMEZONE e SESSIONTIMEZONE). E ciò comporta che "sysdate" potrebbe non coincidere con la data del Sistema Operativo.

anche le frazioni di secondo (per maggiori dettagli leggi il paragrafo dedicato al tipo di dato `TIMESTAMP` più avanti). Secondo quanto trovato in rete ([http://download-east.oracle.com/otn\\_hosted\\_doc/jdeveloper/904preview/jdbc-javadoc/oracle/sql/TIMESTAMP.html](http://download-east.oracle.com/otn_hosted_doc/jdeveloper/904preview/jdbc-javadoc/oracle/sql/TIMESTAMP.html)), questo tipo di dato è storicizzato internamente con 11 bytes come segue:

Byte	Represents
0	Century (119 for 1990)
1	Decade (190 for 1990)
2	Month
3	Day
4	Hour
5	Minute
6	Seconds
7	Nanoseconds
8	Nanoseconds
9	Nanoseconds
10	Nanoseconds

Perché sono necessari 4 bytes in più rispetto a tipo `DATE`? `TIMESTAMP` registra anche le frazioni di secondo che possiamo esprimere fino a 10 digits: da 0 a 9. Poiché con un byte si possono rappresentare 3 digits (1 byte = 8 bit. Questo vuol dire che sono possibili i numeri da 0 a  $2^8=256$ , ovvero 3 digits), sono necessari: 3 bytes per i primi 9 digits più un ulteriore byte per la decima cifra (non possiamo utilizzare meno di un byte, perché 1 byte è l'unità più piccola di gestione).

Detto questo, descriviamo le funzioni prima accennate. La funzione `CURRENT_DATE` restituisce data e tempo corrente nel time zone della sessione (il valore è una data espressa nel calendario Gregoriano). Questo vuol dire che è sensibile al parametro `TIME_ZONE` (espresso come `'[+ | -] hh:mm'`, che rappresenta ore e minuti prima o dopo UTC).

La funzione `CURRENT_TIMESTAMP` restituisce data e frazione di tempo corrente (quindi aggiunge frazioni di secondo rispetto alla precedente funzione `CURRENT_DATE`) nel time zone della sessione (il client), ma come un valore del tipo di dato `TIMESTAMP WITH TIME ZONE`. Anche in questo caso, la funzione è sensibile al parametro `TIME_ZONE`.

La funzione `LOCALTIMESTAMP` restituisce data e frazione di tempo corrente nel time zone della sessione come un valore del tipo di dato `TIMESTAMP`. La differenza tra `LOCALTIMESTAMP` e `CURRENT_TIMESTAMP` è che mentre il primo restituisce un `TIME STAMP`, il secondo restituisce un `TIMESTAMP WITH TIME ZONE`. `TIMESTAMP WITH TIME ZONE` è una variante di `TIMESTAMP`, in cui viene visualizzato il time zone.

### 3. Altre funzioni

Esistono altre funzioni, oltre quelle fin qui discusse, che vale la pena accennare.

#### EXTRAC

Estrae e restituisce il valore di un “datetime” specificato. Il “datetime” può assumere uno dei seguenti valori:

ANNO, MESE, GIORNO, ORA, TIMEZONE\_HOUR, TIMEZONE\_MINUTE, TIMEZONE\_REGION, TIMEZONE\_ABBR.

I valori di TIMEZONE\_REGION, TIMEZONE\_ABBR coincidono con le due colonne della vista V\$TIMEZONE\_NAMES.

La sintassi per tale funzione è:

```
EXTRACT
( { { YEAR | MONTH | DAY | HOUR | MINUTE | SECOND }
  | { TIMEZONE_HOUR | TIMEZONE_MINUTE }
  | { TIMEZONE_REGION | TIMEZONE_ABBR }
  }
  FROM
  { 'datetime_value_expression'
    | 'interval_value_expression'
  }
)
```

Quando si estrae un time zone di una regione o un'abbreviazione, il valore restituito è una stringa contenente l'appropriato time zone o regione. Quando invece si estrae uno degli altri valori, il valore restituito è nel calendario Gregoriano.

#### FROM\_TZ

La funzione FROM\_TZ converte un valore TIMESTAMP in uno TIMESTAMP WITH TIME ZONE. La sintassi è:

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

Dove time\_zone\_value è un stringa di caratteri nel formato 'TZH:TZM' o un'espressione nel formato TZR (Time Zone Region). Come esempio di TZR possiamo dare 'Europe/Rome'. Ricordo che la lista completa delle regioni è possibile prenderla dalla vista V\$TIMEZONE\_REGIONS.

Un esempio chiarificherà molto di più di tante parole.

```
SELECT FROM_TZ(TIMESTAMP '2005-04-03 18:45', '3:00')
```

In pratica, si passa una data (TIMESTAMP), un TIMEZONE ed Oracle restituisce un TIMESTAMP WITH TIME ZONE.

## **TO\_TIMESTAMP, TO\_TIMESTAMP\_TZ**

La funzione TO\_TIMESTAMP converte una stringa di tipo CHAR, VARCHAR2, NCHAR, NVARCHAR2 in un tipo TIMESTAMP. La sintassi è:

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

La sigla fmt, sta per format. Rappresenta infatti il formato con cui si passa la stringa "string". Il parametro opzionale "nlsparam" specifica il linguaggio in cui l'abbreviazione di mese e giorno sono restituiti. L'argomento può avere un'espressione del tipo:

```
'NLS_DATE_LANGUAGE = language'
```

Se questo viene omissso, viene utilizzato il linguaggio di default della sessione.

La funzione TO\_TIMESTAMP\_TZ converte una stringa di tipo CHAR, VARCHAR2, NCHAR, NVARCHAR2 in un tipo TIMESTAMP WITH TIME ZONE. La sintassi per tale funzione è:

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

Il significato dei valori è analogo alla precedente funzione TO\_TIMESTAMP.

## **TO\_YMINTERVAL**

La funzione TO\_YMINTERVAL trasforma una stringa di tipo CHAR, VARCHAR2, NCHAR, NVARCHAR2 in un dato di tipo INTERVAL YEAR TO MONTH. Questo tipo di dato registra un periodo di tempo usando i campi "datetime" YEAR e MONTH. Il formato di INTERVAL YEAR TO MONTH è:

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

Dove "year\_precision" è il numero di digits del campo YEAR. Il valore di default è 2. La sintassi per la funzione TO\_YMINTERVAL è:

```
TO_YMINTERVAL ( strings )
```

Dove "strings" è la stringa di caratteri che deve essere convertita. Facciamo un esempio:

```
SELECT sysdate + TO_YMINTERVAL ('03-11') from dual
```

aggiunge 3 anni ed 11 mesi a sysdate. Vediamone l'output:

```
SQL> select sysdate, sysdate + TO_YMINTERVAL ('03-11') from dual;
```

```
SYSDATE      SYSDATE+T
-----
03-APR-05    03-MAR-09
```

## 4. Il tipo **TIMESTAMP**

Il problema con DATE è sempre stata la granularità. A seguito di due eventi che si succedono, il massimo scarto che è possibile ottenere è 1 secondo. Questo è stato risolto con l'introduzione del tipo di dato **TIMESTAMP** che in qualche modo espande il tipo DATE utilizzando anche le frazioni di secondo. Per convertire un tipo DATE in uno **TIMESTAMP**, si può utilizzare la funzione **CAST**, la cui sintassi è:

```
CAST ({expr | (subquery) | MULTISET(subquery)}
       AS type_name)
```

La funzione **CAST** converte un tipo di dato built-in o un valore collection-typed in un altro tipo di dato built-in o un valore collection-typed.

```
SQL> select sysdate, cast(sysdate as timestamp),
           2   cast(sysdate as timestamp with time zone)
           3   from dual;
```

```
SYSDATE
-----
CAST(SYSDATEASTIMESTAMP)
-----
CAST(SYSDATEASTIMESTAMPWITHTIMEZONE)
-----
04-04-05 11.49.50
04-APR-05 11.49.50.000000 AM
04-APR-05 11.49.50.000000 AM +02:00
```

In questo caso, gli zeri presenti sono dovuti al fatto che il tipo DATE non ha frazioni di secondo e ne viene quindi mostrato il valore di default (6 digits) o se presente, il valore espresso nella variabile **NLS\_TIMESTAMP\_FORMAT**. Come per il tipo DATE, influenzato dalla variabile **NLS\_DATE\_FORMAT** che può

assumere valori del tipo 'DD/MM/YYYY HH24:MI:SS', la variabile NLS\_TIMESTAMP\_FORMAT assume valori del tipo 'DD/MM/YYYY HH24:MI:SSxFF'. La "x" nel formato, fa in modo che nella visualizzazione della data i secondi vengano separati dai frazionali da un punto.

Abbiamo visto che è possibile rappresentare il TIMESTAMP fino a 9 cifre. Per farlo occorre specificare dopo la parola chiave TIMESTAMP e tra parentesi il numero di digits che si desidera visualizzare:

```
SQL> select systimestamp(4) from dual;
```

```
SYSTIMESTAMP(4)
```

```
-----  
13:35:26.5392
```

Un altro modo per modificare in numero di cifre da visualizzare è utilizzare la funzione TO\_CHAR come segue:

```
SQL> select sysdate,systimestamp(4),  
2 to_char(systimestamp , 'hh24:mi:ss.ff5')  
3 from dual;
```

```
SYSTIMESTAMP(4)
```

```
-----  
TO_CHAR(SYSTIMESTA
```

```
-----  
13:32:00.0334
```

```
13:32:00.03345
```

## 5. La data di sistema

Così come per il tipo di dato DATE, esisteva la funzione SYSDATE che restituiva la data corrente ed il tempo, con l'introduzione del tipo di dato TIMESTAMP, esiste una nuova funzione che si chiama SYSTIMESTAMP. Questa restituisce la data di sistema, incluso le frazioni di secondo ed il time zone del sistema su cui i db risiede.

## 6. Conclusioni

i) La nuova funzionalità della data, il `TIMESTAMP`, mostra le date nel time zone della sessione, che è inizializzata dall'ambiente del client. Questo è normalmente ciò che il client si aspetta.

La variabile `TZ`<sup>3</sup> (su Unix) nell'ambiente in cui partono i processi server di Oracle (i così detti processi shadow) determina il "tempo" ricevuto dalle funzioni `SYSDATE` e `SYSTIMESTAMP`. La funzione `SYSDATE` restituisce il tempo locale del server senza l'informazione del time zone. `SYSTIMESTAMP` aggiunge l'offset del time zone del server (se riconosciuto da Oracle) e restituisce `TIMESTAMP WITH TIMEZONE`. Nota che il system time zone del server (settato da `TZ`) non è lo stesso del time zone del database (settato da `CREATE` o `ALTER DATABASE`). Il time zone del database è utilizzato solo per normalizzare i valori `TIMESTAMP WITH LOCAL TIMEZONE` prima di registrarli nel database.

Il time zone della sessione (`SESSIONTIMEZONE`) è inizializzato dalla variabile del client `ORA_SDTZ`, che può assumere i valori `DB_TZ`, `OS_TZ`, un offset del time zone o un nome del time zone. Se il valore è `DB_TZ`, il time zone della sessione sarà settato al time zone del database. Se il suo valore è `OS_TZ`, il time zone della sessione sarà inizializzato al time zone del sistema operativo del client (come definito dalla variabile del client `TZ` su piattaforma Unix o dal Pannello di Controllo su Windows). Questo è il comportamento di default. Se invece è specificato l'offset del time zone o un nome di Regione, allora sarà usato il time zone della sessione.

ii) L'esperienza insegna che per essere sicuri di fare le cose fatte bene conviene seguire i seguenti passi:

1. Controllare ed impostare in modo corretto la data, ed il time zone del sistema operativo. Soprattutto assicurarsi di mettere il controllo di aggiornamento della data di sistema sotto `ntp` (per sistemi Unix),
2. Creare il database facendo attenzione a specificare esplicitamente il corretto time zone. La sintassi è:

```
CREATE DATABASE ... SET TIME_ZONE = 'Europe/Rome'
```

E' importante pensare al corretto time zone quando si crea il db, perché successivamente se si tenta di cambiarlo con il comando

```
ALTER DATABASE SET TIME_ZONE ...
```

si ottiene:

```
SQL> alter database set time_zone = 'Europe/Rome' ;
```

---

<sup>3</sup> Non sono riuscito ancora a trovare documentazione sufficiente per capire il funzionamento della variabile `TZ`.

```
alter database set time_zone = 'Europe/Rome'  
*  
ERROR at line 1:  
ORA-02231: missing or invalid option to ALTER DATABASE
```

Questo a causa di un bug, fissato in Oracle 10g (Nota 230099.1 di Metalink),

3. Eseguire subito dopo la creazione del database, un controllo per confermare il corretto time zone. In particolare simulare una connessione di un client con un diverso time zone. Questo può essere fatto:
  - a. creare una tabella avente un campo con un tipo di dato `TIMESTAMP WITH LOCAL TIME ZONE`,
  - b. modificare (eventualmente) un pc windows su cui è installato un client Oracle e che verrà utilizzato per collegarsi al database, impostando un time zone diverso dal database in modo tale da avere `DBTIMEZONE` diverso da `SESSIONTIMEZONE`,
  - c. utilizzando il client del punto b), popolare la tabella del punto a) con almeno una riga
  - d. fare un `select` sulla tabella popolata al punto c), verificando che se la `select` è eseguita dal client, viene restituita una data nel time zone del client, mentre se fatto il locale sulla macchina (acceduta via ssh o telnet), restituisce una data nel time zone del database.

## 7. Riferimenti

- ✓ Un sito che parla dei time zone:  
<http://www.greenwichmeantime.com/home.htm>
- ✓ Il link dove ho trovato informazioni su quanti bytes Oracle utilizza per storicizzare il tipo di dato `TIMESTAMP`:  
[http://download-east.oracle.com/otn\\_hosted\\_doc/jdeveloper/904preview/jdbc-javadoc/oracle/sql/TIMESTAMP.html](http://download-east.oracle.com/otn_hosted_doc/jdeveloper/904preview/jdbc-javadoc/oracle/sql/TIMESTAMP.html)
- ✓ Bug che non permette la modifica time zone del database:  
Nota 230099.1 di Metalink
- ✓ Modifica per l'estensione dei time zones del database:  
Nota 227334.1 di Metalink

- ✓ La documentazione ufficiale di Oracle:

Il manuale di SQL Reference

[http://download-west.oracle.com/docs/cd/B10501\\_01/server.920/a96540/toc.htm](http://download-west.oracle.com/docs/cd/B10501_01/server.920/a96540/toc.htm)

Il manuale di Developer's Guide

[http://download-west.oracle.com/docs/cd/B10501\\_01/appdev.920/a96590/toc.htm](http://download-west.oracle.com/docs/cd/B10501_01/appdev.920/a96590/toc.htm)

- ✓ Un link che parla del tipo TIMESTAMP:

[http://www.dbasupport.com/oracle/ora9i/DATE\\_TIMESTAMP\\_Datatypes.shtml](http://www.dbasupport.com/oracle/ora9i/DATE_TIMESTAMP_Datatypes.shtml)

- ✓ Un documento sul TIMESTAMP:

[http://www.trivadis.ch/Images/TIMESTAMP\\_e\\_tcm17-7295.pdf](http://www.trivadis.ch/Images/TIMESTAMP_e_tcm17-7295.pdf)