Tipe-tipe Data Oracle 11g

Yuafanda Kholfi Hartono
yuafanda@yahoo.com
http://allofmyjourney.blogspot.com

Datatype (tipe data) adalah klasifikasi atau jenis dari suatu informasi atau data tertentu. Setiap nilai yang dimanipulasi oleh Oracle memiliki sebuah tipe data masing-masing. Tipe data dari sebuah nilai tersebut diasosiasikan dengan nilai properti yang diset tetap. Properti ini menyebabkan nilai-nilai dari satu tipe data diperlakukan berbeda dengan nilai-nilai lain oleh Oracle.

Pendahuluan

Setiap nilai yang dimanipulasi oleh Oracle Database memiliki tipe data masing-masing. Tipe data dari sebuah nilai tersebut diasosiasikan dengan nilai properti yang diset tetap. Properti ini menyebabkan nilai-nilai dari satu tipe data diperlakukan berbeda dengan nilai-nilai lain oleh Oracle. Misalnya, Anda dapat menambahkan besaran nilai pada tipe data NUMBER, tetapi tidak dapat melakukan hal yang sama pada tipe data RAW.

Bila Anda membuat sebuah tabel atau cluster, Anda harus menentukan tipe data untuk masing-masing kolom tersebut. Bila Anda membuat sebuah *procedure* atau *function* yang kemudian akan disimpan, Anda harus menentukan tipe data untuk setiap argumennya. Tipe data ini akan menentukan domain nilai disetiap kolom yang berisi argumen masing-masing yang dapat dimiliki *procedure* atau *function* tersebut. Sebagai contoh, kolom DATE tidak dapat menerima nilai 29 Feb (kecuali untuk tahun kabisat) atau nilai 2 atau 'sepatu'. Setiap nilai akan ditempatkan dalam kolom dengan mengasumsikan tipe data dari kolom tersebut. Misalnya, jika Anda memasukkan '01-JAN-98 'ke dalam kolom DATE, maka Oracle memperlakukan karakter string '01-JAN-98' sebagai nilai DATE setelah memverifikasi karakter string tersebut telah diterjemahkan dalam format tanggal yang valid.

Isi

Oracle Database menyediakan sejumlah *built-in* tipe data serta beberapa kategori untuk jenis yang ditentukan oleh pengguna, yang dapat digunakan sebagai tipe data. Penjelasan dari tipe data Oracle tiap-tiap tipe data akan dijelaskan pada bagian berikut:

Tipe Data Karakter
Tipe data karakter terdiri atas tipe-tipe data CHAR, NCHAR, NVARCHAR2, VARCHAR2, VARCHAR, LONG, RAW dan LONG RAW. Penjelasan dari masing-masing tipe data dijelaskan sebagai berikut :

CHAR
Tipe data CHAR dispesifikasikan dalam karakter string yang memiliki panjang tetap. Oracle memastikan bahwa semua nilai disimpan dalam sebuah kolom CHAR memiliki panjang yang ditentukan oleh ukuran *(size)*. Jika Anda memasukkan nilai yang lebih pendek dari panjang kolom, Oracle akan mengisikan nilai kosong untuk panjang kolom yang tidak terisi nilai. Jika Anda mencoba untuk memasukkan nilai yang terlalu panjang untuk kolom, Oracle akan menampilkan pesan *error*.

Panjang default untuk kolom CHAR adalah 1 byte dan maksimum yang diizinkan adalah 2000 byte. Bila Anda membuat sebuah tabel dengan kolom CHAR, secara default anda diminta untuk menyertakan panjang kolom dalam bytes. Kualifikasi BYTE adalah sama seperti defaultnya. Jika Anda menggunakan kualifikasi CHAR, misalnya CHAR (10 CHAR), maka anda akan diminta menyertakan panjang kolom dalam karakter. Sebuah karakter secara pengertian teknis adalah titik kode dari set karakter database. Ukurannya dapat berkisar dari 1 byte sampai 4 byte, tergantung pada pengaturan karakter database. Kualifikasi BYTE dan CHAR ditentukan oleh parameter NLS_LENGTH_SEMANTICS, yang memiliki default semantik byte. Untuk alasan performa, Oracle menyarankan Anda menggunakan parameter NLS_LENGTH_SEMANTICS untuk mengatur panjang semantik.

Untuk memastikan konversi yang tepat data antara database dengan set karakter yang berbeda, Anda harus memastikan bahwa data CHAR terdiri dari *well-formed* string.

NCHAR
Tipe data NCHAR adalah tipe data *Unicode-only*. Bila Anda membuat sebuah tabel dengan kolom NCHAR, Anda akan diminta menentukan panjang kolom dalam karakter. Anda mendefinisikan karakter nasional saat Anda membuat (*create*) database Anda.

Panjang maksimum kolom ditentukan oleh definisi set karakter nasional. Spesifikasi Lebar tipe data karakter NCHAR mengacu pada jumlah karakter. Ukuran kolom maksimum yang diizinkan adalah 2000 byte.

Jika Anda memasukkan nilai yang lebih pendek dari panjang kolom, Oracle akan mengisikan nilai kosong untuk panjang kolom yang tidak terisi nilai. Anda tidak dapat memasukkan nilai CHAR menjadi kolom NCHAR, Anda juga tidak bisa memasukkan nilai NCHAR ke kolom CHAR.

Contoh berikut membandingkan kolom translated_description dari tabel pm.product_descriptions dengan string karakter nasional:

```
SELECT translated_description FROM product_descriptions
   WHERE translated_name = N'LCD Monitor 11/PM ';
```

NVARCHAR2
Tipe data NVARCHAR2 adalah tipe data *Unicode-only*. Bila Anda membuat sebuah tabel dengan kolom NVARCHAR2, anda akan diminta menyertakan jumlah maksimal karakter yang dapat diisikan kedalamnya. Oracle kemudian menyimpan setiap nilai dalam kolom persis seperti yang Anda tentukan itu, asalkan nilai tidak melebihi panjang maksimum kolom.

Panjang maksimum kolom ditentukan oleh definisi set karakter nasional. Spesifikasi Lebar karakter tipe data NVARCHAR2 mengacu pada jumlah karakter. Ukuran kolom maksimum yang diizinkan adalah 4000 byte.

VARCHAR2
Tipe data VARCHAR2 menetapkan string karakter variabel-panjang. Ketika Anda membuat kolom VARCHAR2, anda akan diminta menyertakan jumlah maksimal byte atau karakter data yang dapat diisikan kedalamnya. Oracle kemudian menyimpan setiap nilai dalam kolom persis seperti yang Anda tentukan itu, asalkan nilai tidak melebihi panjang maksimum kolom tentang kolom. Jika Anda mencoba untuk memasukkan nilai yang melebihi panjang yang ditentukan, maka Oracle akan menampilkan pesan *error*.

Anda harus menentukan panjang maksimum untuk kolom VARCHAR2, minimal 1 byte, meskipun sebenarnya string yang disimpan diperkenankan menjadi string dengan panjang nol (''). Dapat juga menggunakan kualifikasi CHAR, misalnya untuk VARCHAR2 (10 CHAR), untuk memberikan panjang maksimum karakter bukan byte. Sebuah karakter secara teknis titik kode dari set karakter database. Anda dapat menggunakan kualifikasi BYTE, misalnya untuk VARCHAR2 (10 BYTE), secara eksplisit untuk memberikan panjang maksimum dalam bytes. Jika tidak ada kualifikasi eksplisit dimasukkan dalam definisi kolom atau atribut jika *object* database dengan kolom atau atribut dibuat, maka semantik panjang ditentukan oleh nilai parameter NLS_LENGTH_SEMANTICS dari *session create object*. Terlepas dari panjang maksimum dalam karakter, panjang data VARCHAR2 tidak dapat melebihi 4000 byte. Oracle membandingkan nilai VARCHAR2 menggunakan perbandingan semantik *nonpadded*.

Untuk memastikan konversi yang tepat data antara database dengan set karakter yang berbeda, Anda harus memastikan bahwa data VARCHAR2 terdiri dari string *well-formed*.

VARCHAR
Jangan menggunakan tipe data VARCHAR. Gunakan VARCHAR2 sebagai gantinya. Meskipun tipe data VARCHAR saat ini identik dengan VARCHAR2, tipe data VARCHAR dijadwalkan akan didefinisikan ulang sebagai tipe data terpisah yang digunakan untuk string karakter variabel-panjang dibandingkan dengan perbandingan semantik yang berbeda.

LONG
Jangan membuat tabel dengan menggunakan kolom LONG. Gunakan kolom LOB (CLOB, NCLOB, BLOB) sebagai gantinya. kolom LONG didukung hanya untuk kompatibilitas.

kolom LONG menyimpan string karakter yang mengandung variabel-panjang sampai dengan 2 gigabyte -1 atau 2^{31}-1 byte. Kolom LONG memiliki banyak karakteristik kolom VARCHAR2. Anda dapat menggunakan kolom LONG untuk menyimpan string teks panjang. Panjang nilai LONG mungkin dibatasi oleh memori yang tersedia pada komputer Anda.

Oracle juga merekomendasikan Anda untuk mengkonversi kolom LONG dari database Anda ke kolom LOB. Kolom LOB memiliki pembatasan jauh lebih sedikit daripada kolom LONG, fungsi LOB ditingkatkan di setiap rilis, sedangkan fungsi LONG telah statis untuk beberapa rilis terakhir.

Anda dapat kolom referensi LONG dalam laporan SQL di tempat-tempat:
- Daftar SELECT
- Klausa SET dari UPDATE *statement*
- Klausa VALUE dari INSERT *statement*

Penggunaan nilai LONG dikenakan pembatasan sebagai berikut :

- Tabel hanya dapat berisi satu kolom LONG.
- Anda tidak dapat membuat sebuah tipe objek dengan atribut LONG.
- Kolom LONG tidak dapat muncul dalam klausa WHERE atau dalam kendala integritas (kecuali bahwa mereka dapat muncul dalam *constraint* NULL dan NOT NULL).
- Kolom LONG tidak dapat diindeks.
- Data LONG tidak dapat ditentukan dalam ekspresi reguler.
- *Function* yang tersimpan tidak dapat mengembalikan nilai LONG
- Anda dapat mendeklarasikan variabel atau argumen dari unit / program PL SQL menggunakan tipe data LONG, Namun anda tidak bisa kemudian memanggil unit program dari SQL tersebut.
- Dalam SQL *statement* tunggal, semua kolom LONG, tabel yang diperbarui, dan tabel terkunci harus terletak pada database yang sama.
- Kolom LONG dan LONG RAW tidak dapat digunakan dalam SQL *statement* yang didistribusikan dan tidak bisa direplikasi.
- Jika sebuah tabel memiliki kolom LONG dan LOB, maka Anda tidak dapat mengikat lebih dari 4000 byte data untuk kedua kolom LONG dan LOB dalam SQL *statement* yang sama. Namun, Anda dapat mengikat lebih dari 4000 byte data ke salah satu LONG atau kolom LOB.

Selain itu, kolom LONG tidak dapat muncul dalam bagian-bagian SQL *statement*:

- Klausa GROUP BY, ORDER BY, atau klausa CONNECT BY atau dengan operator dalam *statement* SELECT DISTINCT
- Operator UNIQUE dari sebuah SELECT *statement*
- Daftar kolom dalam CREATE CLUSTER *statement*
- Klausa CLUSTER pada CREATE MATERIALIED VIEW *statement*
- *Function*, ekspresi, atau kondisi SQL built-in
- Daftar query SELECT yang mengandung klausa GROUP BY
- Daftar dari subqueries SELECT atau pertanyaan dikombinasikan oleh UNION, INTERSECT, atau operator MINUS
- Daftar SELECT dari *statement* CREATE TABLE ... AS SELECT
- ALTER TABLE ... MOVE *statement*
- Daftar SELECT di subqueries dalam INSERT *statement*

RAW dan LONG RAW

Tipe data RAW dan LONG RAW menyimpan data yang tidak secara eksplisit dikonversi oleh Oracle Database ketika memindahkan data antara sistem yang berbeda. Tipe data ini dimaksudkan untuk data biner atau *string byte*. Sebagai contoh, Anda dapat menggunakan LONG RAW untuk menyimpan grafik, *sound*, dokumen, atau *array* data biner, yang penafsirannya tergantung pada penggunaan masing-masing.

Oracle sangat menyarankan agar Anda mengubah kolom LONG RAW ke kolom biner LOB (BLOB). Kolom LOB memiliki pembatasan jauh lebih sedikit daripada kolom LONG. RAW adalah tipe data variabel-panjang seperti VARCHAR2, kecuali bahwa *Oracle Net* (yang menghubungkan sesi pengguna ke *instance*) dan *Oracle import and export utilities* tidak melakukan konversi karakter selama transmisi data RAW atau LONG RAW. Sebaliknya, *Oracle Net* dan *Oracle import and export utilities* secara otomatis mengkonversi CHAR, VARCHAR2, dan data LONG dari database karakter set ke karakter set *user session*. Jika terdapat dua set

karakter yang berbeda, Anda dapat mengatur karakter set sesi pengguna dengan parameter NLS_LANGUAGE dari *statement* ALTER SESSION.

Ketika Oracle secara otomatis mengkonversi data RAW atau LONG RAW ke dan dari data CHAR, data biner direpresentasikan dalam bentuk heksadesimal, dengan satu karakter heksadesimal yang mewakili setiap empat bit data RAW. Sebagai contoh, satu byte data RAW dengan bit 11001011 ditampilkan dan dimasukkan sebagai CB.

Tipe Data Numerik
Tipe data Numerik terdiri dari tipe-tipe data NUMBER, FLOAT, BINARY FLOAT dan BINARY DOUBLE

NUMBER
Tipe data NUMBER terdiri dari angka tetap nol serta positif dan negatif dengan nilai absolut dari $1,0 \times 10^{-130}$ untuk tetapi tidak lebih dari $1,0 \times 10^{126}$. Jika Anda menetapkan sebuah ekspresi aritmatika yang nilainya memiliki nilai absolut lebih besar dari atau sama dengan $1,0 \times 10.^{126}$, maka Oracle akan menampilkan pesan *error*. Setiap nilai NUMBER membutuhkan 1sampai dengan 22 byte.

Anda dapat menentukan *fixed-point* NUMBER dengan menggunakan formula berikut:

NUMBER (p, s)

dimana:

- p adalah presisi, atau jumlah maksimum dari digit desimal yang signifikan, di mana yang paling signifikan adalah digit paling kiri nol digit, dan yang paling signifikan adalah digit angka kanan yang paling dikenal. Oracle menjamin portabilitas angka dengan presisi hingga 20 digit basis-100, yang setara dengan 39 atau 40 digit desimal tergantung pada posisi titik desimal.
- s adalah skala, atau jumlah digit dari titik desimal ke angka paling signifikan. Skala dapat berkisar dari -84 sampai 127.

skala Positif adalah jumlah digit yang signifikan ke kanan titik desimal dengan dan termasuk angka paling signifikan.
skala negatif adalah jumlah digit yang signifikan di sebelah kiri titik desimal, untuk tetapi tidak termasuk yang paling signifikan digit. Untuk skala negatif paling signifikan digit di sisi kiri dari titik desimal, karena data aktual adalah dibulatkan ke nomor tertentu tempat di sebelah kiri titik desimal. Misalnya, spesifikasi (10, -2) berarti bulat untuk ratusan.

Skala dapat lebih besar dari presisi, paling sering ketika notasi e digunakan. Ketika skala lebih besar dari presisi, ketepatan menentukan jumlah maksimum digit yang signifikan ke kanan titik desimal. Sebagai contoh, kolom didefinisikan sebagai NOMOR (4,5) memerlukan nol untuk digit pertama setelah titik desimal dan putaran semua nilai masa lalu digit kelima setelah titik desimal.

Ini adalah praktik yang baik untuk menentukan skala dan ketepatan kolom nomor *fixed-point* untuk memeriksa integritas tambahan input. Menentukan skala dan presisi tidak membuat semua nilai yang diinputkan harus dengan panjang tetap. Jika nilai melebihi presisi, Oracle akan menampilkan pesan *error*. Jika nilai melebihi skala, maka Oracle akan mengulanginya kembali.

Subtipe data NUMBER

Anda dapat menggunakan NUMBER subtipe berikut untuk kompatibilitas dengan ANSI / ISO dan jenis IBM atau ketika Anda ingin nama yang lebih deskriptif:

DEC
DECIMAL
DOUBLE PRECISION
FLOAT
INT
INTEGER
NUMERIC
REAL
SMALLINT

Gunakan subtipe data DEC, DECIMAL, dan NUMERIC untuk menyatakan nomor *fixed-point* dengan presisi maksimum 38 digit desimal.

Gunakan subtipe data DOUBLE PRECISION dan FLOAT untuk menyatakan angka *floating-point* dengan presisi maksimum 126 digit biner, yang secara kasar setara dengan 38 digit desimal Atau,. gunakan subtipe data REAL untuk menyatakan angka *floating-point* dengan presisi maksimum 63 biner digit, yang secara kasar setara dengan 18 digit desimal.

Gunakan subtipe data INTEGER, INT, dan SMALLINT untuk menyatakan bilangan bulat dengan presisi maksimum 38 digit desimal.

FLOAT
Tipe data FLOAT adalah subtipe dari NUMBER. Hal ini dapat ditentukan dengan atau tanpa presisi, yang memiliki definisi yang sama dengan yang dimiliki tipe data NUMBER dan dapat berkisar dari 1 sampai 126. Skala yang belum bisa ditentukan, tetapi diinterpretasikan dari data. Setiap nilai FLOAT membutuhkan 1 sampai dengan 22 byte.

Untuk mengkonversi dari biner ke desimal presisi, kalikan *n* dengan 0,30103. Untuk mengkonversi dari desimal ke biner presisi, kalikan presisi desimal dengan 3,32193. Maksimum 126 digit presisi biner secara kasar setara dengan 38 digit desimal presisi.

Perbedaan antara NUMBER dan FLOAT yang terbaik akan diilustrasikan dengan contoh. Pada contoh berikut ini nilai yang sama dimasukkan ke dalam kolom NUMBER dan FLOAT:

```
CREATE TABLE test (col1 NOMOR (5,2), col2 FLOAT (5));

INSERT INTO tes VALUES (1,23, 1,23);
INSERT INTO tes VALUES (7,89, 7,89);
INSERT INTO tes VALUES (12,79, 12,79);
INSERT INTO tes VALUES (123,45, 123,45);
SELECT * FROM test;
     Col1          col2
----------    ----------
     1.23           1.2
     7.89           7.9
    12.79            13
```

123.45 120

Dalam contoh diatas, nilai FLOAT tidak dapat melebihi 5 digit biner. Angka desimal terbesar yang dapat diwakili oleh 5 digit biner 31. Baris terakhir berisi nilai desimal yang melebihi 31. Oleh karena itu, nilai FLOAT harus terpotong sehingga angka yang signifikan tidak memerlukan lebih dari 5 digit biner. Jadi 123,45 adalah dibulatkan ke 120, yang hanya dua angka desimal yang signifikan, dan hanya membutuhkan 4 digit biner.

Oracle Database menggunakan tipe data FLOAT secara internal saat mengkonversi data FLOAT ANSI. Oracle FLOAT tersedia bagi Anda untuk digunakan, tetapi Oracle menyarankan Anda menggunakan tipe data BINARY_FLOAT dan BINARY_DOUBLE, karena mereka lebih kuat. Lihat pembahasan "*Floating-Point Number*" untuk informasi lebih lanjut.

Floating-Point Numbers

Floating-point number dapat memiliki titik desimal dimana saja, dari digit pertama sampai dengan yang terakhir, atau bisa juga tidak memiliki titik desimal sama sekali. Eksponen ini dapat digunakan dengan menempatkannya setelah angka tertentu untuk meningkatkan jangkauan, misalnya, 1,777 e^{-20}. Sebuah nilai skala tidak berlaku untuk *floating-point number*, karena jumlah angka yang dapat muncul setelah titik desimal tidak dibatasi.

Binary floating-point number berbeda dari tipe data NUMBER pada saat nilai-nilainya disimpan secara internal oleh Oracle Database. Untuk tipe data NUMBER, nilai yang disimpan menggunakan presisi desimal. Semua literal yang berada dalam jangkauan dan presisi yang disuport oleh NUMBER disimpan sebagai NUMBER. Literal disimpan justru karena disajikan menggunakan presisi desimal (angka 0 sampai 9). *Binary floating-point number* tersebut disimpan dengan menggunakan presisi biner (angka 0 dan 1). Dalam hal ini, *storage* tidak bisa mewakili semua nilai menggunakan desimal dengan presisi yang tepat. Seringkali terdapat kesalahan yang terjadi ketika mengkonversi nilai dari presisi desimal ke biner dibatalkan ketika nilai tersebut dikonversi kembali dari presisi biner ke desimal.

Oracle Database menyediakan dua tipe data numerik secara eksklusif untuk *floating-point number*:

BINARY_FLOAT
BINARY_FLOAT adalah 32-bit, satu-presisi tipe data *floating-point number*. Setiap nilai BINARY_FLOAT memerlukan 5 byte, termasuk panjang byte.

BINARY_DOUBLE
BINARY_DOUBLE adalah 64-bit, ganda tipe data presisi *floating-point number*. Setiap nilai BINARY_DOUBLE memerlukan 9 byte, termasuk panjang byte.

Dalam kolom NUMBER, *floating-point number* memiliki presisi desimal. Dalam kolom BINARY_FLOAT atau BINARY_DOUBLE, *floating-point number* memiliki presisi biner. *Binary floating-point numbers* mendukung nilai-nilai khusus tak terhingga dan NaN (bukan angka).

Tipe Data Tanggal / Waktu
Tipe data tanggal/waktu terdiri dari tipe data DATE, TIMESTAMP, TIMESTAMP WITH TIME ZONE, dan TIMESTAMP WITH LOCAL TIME ZONE. Nilai tipe data datetime kadang-kadang disebut *datetimes*. Sedangkan tipe data interval terdiri dari INTERVAL YEAR TO

MONTH dan INTERVAL DAY TO SECOND. Nilai tipe data interval kadang-kadang disebut *intervals*. Untuk informasi tentang mengekspresikan nilai-nilai datetime dan interval sebagai literal, lihat "Literal Datetime" dan "Interval Literal".

Baik *datetimes* dan interval terdiri dari *fields*. Nilai *field* ini menentukan nilai tipe data tersebut. Untuk menghindari hasil yang tidak terduga dalam operasi DML Anda pada data datetime, Anda dapat memverifikasi database dan zona waktu dengan query sesi built-in fungsi SQL DBTIMEZONE dan SESSIONTIMEZONE. Jika zona waktu yang diatur secara manual, maka Oracle Database menggunakan zona waktu sistem operasi secara default. Jika zona waktu sistem operasi bukan Oracle zona waktu yang valid, maka Oracle menggunakan UTC sebagai nilai default.

DATE
Tipe data DATE menyimpan informasi tanggal dan waktu. Meskipun tanggal dan informasi waktu dapat diwakili di kedua tipe data karakter dan number, datatype DATE memiliki sifat yang terkait khusus. Untuk setiap nilai DATE, Oracle menyimpan informasi berikut: abad, tahun, bulan, tanggal, jam, menit, dan detik.

Anda dapat menetapkan nilai DATE sebagai literal, atau Anda dapat mengkonversi karakter atau nilai numerik ke nilai tanggal dengan fungsi TO_DATE. Contoh untuk mengekspresikan nilai-nilai DATE dengan kedua cara ini, lihat "Datetime Literal".

Menggunakan Kalender Julian
Kalender Julian adalah jumlah hari sejak 1 Januari 4712 SM. hari Julian digunakan sebagai referensi umum. Anda dapat menggunakan model format tanggal "J" dengan fungsi tanggal TO_DATE dan TO_CHAR untuk mengkonversi antara nilai-nilai Oracle DATE dengan nilai setaranya dalam kalender Julian.

Nilai default Tanggal ditentukan sebagai berikut:

- Tahun adalah tahun berjalan, seperti nilai yang dikeluarkan oleh SYSDATE.
- Bulan adalah bulan berjalan, seperti nilai yang dikeluarkan oleh SYSDATE.
- Hari adalah 01 (hari pertama bulan).
- Jam, menit, dan detik semua 0.

Nilai-nilai default yang digunakan dalam sebuah query yang meminta nilai-nilai tanggal dimana tanggal itu sendiri tidak ditentukan, seperti dalam contoh berikut, yang diterbitkan pada bulan Mei:

```
SELECT TO_DATE ('2005 ',' YYYY ') FROM DUAL;

TO_DATE ('
---------
01-MEI-05
```

Contoh *statement* berikut ini mengembalikan nilai setara dari Kalender Julian 1 Januari 1997:

```
SELECT TO_CHAR (TO_DATE ('01-01-1997 ',' MM-DD-YYYY '),' J ')
  FROM DUAL;

TO_CHAR
--------
2450450
```

TIMESTAMP
Tipe data TIMESTAMP merupakan perpanjangan dari tipe data DATE. Tipe data ini menyimpan tahun, bulan, dan hari tipe data DATE, ditambah nilai-nilai jam, menit, dan detik. Tipe data ini berguna untuk menyimpan nilai waktu yang tepat. Rumus dari tipe data TIMESTAMP adalah sebagai berikut:

TIMESTAMP [(fractional_seconds_precision)]

dimana fractional_seconds_precision opsional menentukan jumlah digit yang disimpan Oracle di bagian pecahan dari field DETIK datetime tersebut. Bila Anda membuat kolom dengan tipe data ini, nilainya bisa menjadi nomor dalam rentang 0 sampai 9. Dengan nilai default 6.

TIMESTAMP WITH TIME ZONE
TIMESTAMP WITH TIME ZONE adalah varian dari TIMESTAMP yang meliputi zona waktu nama wilayah atau zona waktu aa offset dalam nilainya. Zona waktu offset adalah perbedaan (dalam jam dan menit) antara waktu setempat dan UTC (Coordinated Universal Time-dahulu Greenwich Mean Time). Datatype ini berguna untuk mengumpulkan dan mengevaluasi informasi terkini di seluruh wilayah geografis.

Rumus tipe data TIMESTAMP WITH TIME ZONE adalah sebagai berikut:

TIMESTAMP [(fractional_seconds_precision)] WITH TIME ZONE

dimana fractional_seconds_precision opsional menentukan jumlah digit yang disimpan Oracle di bagian pecahan dari field DETIK datetime tersebut. Bila Anda membuat kolom dengan tipe data ini, nilainya bisa menjadi nomor dalam rentang 0 sampai 9. Dengan nilai default 6.

TIMESTAMP WITH LOCAL TIME ZONE
TIMESTAMP WITH LOCAL TIME ZONE adalah varian lain dari TIMESTAMP yang meliputi zona waktu yang disajikan pada nilainya. Tipe data ini berbeda dari TIMESTAMP WITH TIME ZONE dalam hal data yang tersimpan dalam database adalah dinormalisasi dengan zona waktu database, dan zona waktu offset tidak disimpan sebagai bagian dari kolom data. Bila pengguna mengambil data tersebut, Oracle kembali dalam zona waktu lokal pengguna sesi. Zona waktu offset adalah perbedaan (dalam jam dan menit) antara waktu setempat dan UTC (*Coordinated Universal Time*-dahulu *Greenwich Mean Time*). Tipe data ini berguna untuk menampilkan informasi terkini dalam zona waktu dari sistem klien dalam aplikasi dua-tier.

Rumus tipe data TIMESTAMP WITH LOCAL TIME ZONE adalah sebagai berikut:

TIMESTAMP [(fractional_seconds_precision)] WITH LOCAL TIME ZONE

dimana fractional_seconds_precision opsional menentukan jumlah digit yang disimpan Oracle di bagian pecahan dari field DETIK datetime tersebut. Bila Anda membuat kolom dengan tipe data ini, nilainya bisa menjadi nomor dalam rentang 0 sampai 9. Dengan nilai default 6.

INTERVAL YEAR TO MONTH
INTERVAL YEAR TO MONTH menyimpan periode waktu menggunakan field datetime YEAR dan MONTH. Tipe data ini berguna untuk mewakili perbedaan antara dua nilai datetime ketika hanya nilai tahun dan bulan yang signifikan.

Rumus INTERVAL YEAR TO MONTH adalah sebagai berikut:

INTERVAL YEAR [(year_precision)] TO MONTH

dimana year_precision adalah jumlah digit pada field datetime YEAR. Nilai default dari year_precision adalah 2.

INTERVAL DAY TO SECOND
INTERVAL DAY TO SECOND menyimpan periode waktu dalam hari, jam, menit, dan detik. Tipe data ini berguna untuk mewakili perbedaan yang tepat antara dua nilai datetime.

Rumus tipe data ini adalah sebagai berikut:

INTERVAL DAY[(day_precision)]
  TO SECOND [(fractional_seconds_precision)]

dimana
- day_precision adalah jumlah digit pada field datetime DAY. Nilai yang dapat diterima adalah 0 sampai 9. Dengan nilai default 2.
- fractional_seconds_precision adalah jumlah digit di bagian pecahan dari bidang datetime DETIK. Nilai yang dapat diterima adalah 0 sampai 9. Dengan nilai default 6.

Datetime aritmatika / Interval
Anda dapat melakukan sejumlah operasi aritmatika pada tipe data tanggal (DATE), timestamp (TIMESTAMP, TIMESTAMP WITH TIME ZONE, dan TIMESTAMP WITH LOCAL TIME ZONE) dan data interval(INTERVAL YEAR TO MONTH dan INTERVAL DAY TO SECOND). Oracle menghitung hasilnya berdasarkan aturan berikut:
- Anda dapat menggunakan konstanta NUMBER dalam operasi aritmatika pada nilai-nilai tanggal dan timestamp, tapi tidak nilai-nilai interval. Oracle secara internal mengkonversi nilai timestamp untuk tanggal dan menafsirkan nilai-nilai konstanta NUMBER dalam datetime aritmatika dan ekspresi interval sebagai jumlah hari. Sebagai contoh, SYSDATE + 1 adalah besok. SYSDATE - 7 adalah satu minggu yang lalu. SYSDATE + (10/1440) adalah sepuluh menit dari sekarang. Mengurangkan SYSDATE dengan kolom *hire_date* karyawan dari tabel *test* akan menghasilkan jumlah hari sejak masing-masing karyawan dipekerjakan. Anda tidak dapat mengalikan atau membagi nilai tanggal atau timestamp.
- Oracle secara implisit mengubah operan BINARY_FLOAT dan BINARY_DOUBLE ke NUMBER.
- Setiap nilai DATE mengandung komponen waktu, dan hasil dari operasi manipulasi tanggal termasuk fraksi didalamnya. Fraksi berarti sebagian dari satu hari. Misalnya, 1,5 hari adalah 36 jam. Fraksi ini juga dikembalikan oleh fungsi Oracle built-in untuk operasi umum pada data DATE. Sebagai contoh, fungsi MONTHS_BETWEEN mengembalikan jumlah bulan antara dua tanggal. Bagian pecahan dari hasil yang merupakan bagian dari satu bulan yang terdiri dari 31 hari.
- Jika salah satu operan adalah nilai DATE atau nilai numerik, baik yang mengandung zona waktu atau pecahan komponen detik, maka:
    - Oracle secara implisit akan mengubah operan lain untuk data DATE. Pengecualian adalah perkalian dari nilai numerik interval, yang akan memiliki nilai kembali berupa interval.
    - Jika operan lain memiliki nilai zona waktu, maka Oracle menggunakan zona waktu sesi pada nilai kembali.
    - Jika operan lain memiliki nilai pecahan detik, maka nilai pecahan detik dihilangkan.

- Bila Anda lulus timestamp, interval, atau nilai numerik untuk built-in fungsi yang dirancang hanya untuk datatype DATE, Oracle secara implisit mengkonversi nilai non-DATE ke nilai DATE.
- Ketika perhitungan interval mengembalikan nilai datetime, hasilnya harus nilai datetime aktual atau database akan menampilkan pesan *error*. Sebagai contoh, berikutnya dua *statement* yang akan menghasilkan pesan *error*:

```
SELECT TO_DATE ('31-AUG-2004','DD-MON-YYYY') + TO_YMINTERVAL ('0 -1') FROM
DUAL;

SELECT TO_DATE ('29-FEB-2004',' DD-MON-YYYY') + TO_YMINTERVAL ('1 - 0') FROM
DUAL;
```

Statement pertama akan gagal karena menambahkan satu bulan ke bulan 31 hari akan menghasilkan 31 September yang bukan merupakan tanggal yang valid. *Statement* kedua gagal karena menambah satu tahun ke tanggal yang hanya ada setiap empat tahun sehingga dianggap tidak valid. Namun, *statement* berikutnya berhasil, karena menambahkan empat tahun ke tanggal 29 Februari sehingga dianggap valid:

```
SELECT TO_DATE ('29-FEB-2004 ',' DD-MON-YYYY ') + TO_YMINTERVAL (-0 '4') FROM
DUAL;

TO_DATE ('
---------
29-FEB-08
```

- Oracle melakukan semua aritmatika timestamp dalam waktu UTC. Untuk TIMESTAMP WITH LOCAL TIME ZONE, Oracle mengkonversi nilai datetime dari zona waktu UTC dan database untuk mengubah kembali ke zona waktu database setelah melakukan aritmetik. Untuk TIMESTAMP WITH TIME ZONE, nilai datetime selalu di UTC, sehingga konversi tidak diperlukan.

Tipe data Large Object (LOB)

Tipe-tipe data *built-in* LOB seperti BLOB, CLOB, dan NCLOB (disimpan secara internal) dan BFILE (disimpan secara eksternal) dapat menyimpan data yang besar dan tidak terstruktur seperti teks, gambar, video, dan data spasial. Ukuran BLOB, CLOB, dan data NCLOB bisa sampai (2^{32}-1 bytes) * (nilai parameter chunk pada media penyimpanan LOB). Jika tablespace dalam database Anda adalah dari ukuran blok standar, dan jika Anda telah menggunakan nilai default parameter chunk penyimpanan LOB saat membuat kolom LOB, maka ini setara dengan (2^{32}-1 bytes) * (database block size) . Ukuran data BFILE dapat mencapai 2^{64}-1 byte, meskipun sistem operasi Anda dapat mengenakan pembatasan maksimum.

Ketika membuat sebuah tabel, anda dapat menentukan tablespace yang berbeda dan karakteristik penyimpanan untuk kolom LOB atau atribut objek LOB dari yang ditentukan untuk tabel.

CLOB, NCLOB, dan nilai-nilai BLOB sampai dengan kira-kira 4000 byte disimpan inline jika Anda mengaktifkan storage pada baris pada saat kolom LOB dibuat. Sedangkan apabila data LOB lebih besar dari 4000 byte maka akan disimpan eksternal.

Kolom LOB mengandung LOB *locator* yang dapat mengacu pada nilai-nilai LOB internal

(dalam database) atau eksternal (di luar database). Memilih LOB dari tabel akan mengembalikan LOB *locator* dan bukan nilai LOB keseluruhan. Operasi paket DBMS_LOB dan *Oracle Call Interface* (OCI) di LOB dilakukan melalui *locator*.

LOB mirip dengan tipe LONG RAW dan LONG, namun berbeda dalam hal-hal berikut:

- LOB bisa menjadi atribut dari suatu tipe objek (*datatype user-defined*).
- LOB *locator* disimpan dalam kolom tabel, baik dengan atau tanpa nilai LOB sebenarnya. BLOB, NCLOB, dan nilai-nilai CLOB dapat disimpan dalam tablespace terpisah. BFILE data disimpan dalam file eksternal pada server.
- Bila Anda mengakses kolom LOB, *locator* dikembalikan.
- Sebuah LOB bisa berukuran sampai (2^{32}-1 bytes) * (database block size). Data BFILE bisa berukuran sampai 2^{64}-1 byte, meskipun sistem operasi Anda dapat mengenakan pembatasan maksimum.
- Anda dapat menentukan lebih dari satu kolom LOB dalam sebuah tabel.
- Dengan pengecualian NCLOB, Anda dapat menentukan satu atau lebih atribut LOB dalam suatu objek.
- Anda dapat mendeklarasikan variabel mengikat (*bind variables*) LOB.
- Anda dapat memilih kolom LOB dan atribut LOB.
- Anda dapat menyisipkan baris baru atau memperbarui suatu baris yang sudah ada yang berisi satu atau lebih kolom LOB atau objek dengan satu atau lebih atribut LOB. Dalam operasi update, anda dapat mengatur nilai LOB internal untuk NULL, kosong, atau mengganti seluruh LOB dengan data. Anda dapat mengatur BFILE ke NULL atau membuat agar menunjuk ke sebuah file yang berbeda.
- Anda dapat memperbarui simpang (*interception*) baris-kolom LOB atau atribut LOB dengan simpangan baris-kolom LOB atau atribut LOB yang lain.
- Anda dapat menghapus sebuah baris yang berisi kolom LOB atau atribut LOB dan dengan demikian juga menghapus nilai LOB. Untuk BFILEs, file sistem operasi yang sebenarnya tidak dihapus.

Anda dapat mengakses dan mengisi baris dari kolom LOB inline (kolom LOB disimpan dalam database) atau atribut LOB (atribut dari sebuah kolom tipe objek yang disimpan dalam database) hanya dengan mengeluarkan pernyataan INSERT atau UPDATE.

BFILE

Tipe data BFILE memungkinkan akses ke file LOB biner yang disimpan dalam sistem file di luar Oracle Database. Sebuah kolom atau atribut BFILE disimpan dalam locator BFILE, yang berfungsi sebagai pointer ke file biner pada sistem file server. Locator mempertahankan nama direktori dan nama file.

Anda dapat mengubah nama file dan path dari BFILE tanpa mempengaruhi tabel base dengan menggunakan fungsi BFILENAME. Binary file LOB tidak berpartisipasi dalam transaksi dan tidak dapat direcovery (dipulihkan). Sebaliknya, sistem operasi yang mendasarinya menyediakan integritas file dan daya tahan. Data BFILE bisa sampai 2^{64}-1 byte, meskipun sistem operasi Anda dapat mengenakan pembatasan maksimum.

Database administrator harus memastikan bahwa file eksternal ada dan bahwa sistem Oracle proses memiliki izin operasi *read* pada file tersebut.

Tipe data BFILE memungkinkan dukungan *read-only* file biner yang berukuran besar. Anda tidak dapat mengubah atau mereplikasi file tersebut. Oracle menyediakan API untuk mengakses

data file. Interface utama yang Anda gunakan untuk mengakses file data adalah *package* DBMS_LOB dan *Oracle Call Interface* (OCI).

BLOB
Tipe data BLOB tersusun atas *large object* biner yang tidak terstruktur. Objek BLOB dapat dianggap sebagai bitstreams dengan karakter tanpa set semantik. Objek BLOB dapat menyimpan data biner sampai dengan (4 gigabyte -1) * (nilai parameter CHUNK LOB *storage*). Jika tablespace dalam database Anda adalah ukuran blok standar, dan jika Anda telah menggunakan nilai default parameter chunk LOB *storage* saat membuat kolom LOB, setara dengan (4 gigabyte - 1) * (*database block size*) .

Objek BLOB memiliki dukungan penuh terhadap transaksional. Perubahan yang dilakukan melalui SQL, *package* DBMS_LOB, atau *Oracle Call Interface* (OCI) berpartisipasi penuh dalam transaksi. Manipulasi nilai BLOB dapat dilakukan dan dibatalkan. Namun, Anda tidak dapat menyimpan *locator* BLOB dalam variabel PL / SQL atau OCI dalam satu transaksi dan kemudian menggunakannya dalam transaksi atau *session* lain.

CLOB
Tipe data CLOB terdiri atas *single-byte* dan *multibyte* data karakter. Keduanya memiliki lebar tetap dan didukung seting karakter variabel-lebar, selain itu keduanya juga menggunakan karakter set database. Objek CLOB dapat menyimpan sampai dengan (4 gigabyte -1) * (nilai parameter CHUNK LOB *storage*) dari karakter data. Jika tablespace dalam database Anda memiliki ukuran blok standar, dan jika Anda telah menggunakan nilai default parameter chunk LOB *storage* saat membuat kolom LOB, maka ini setara dengan (4 gigabyte - 1) * (*database block size*) .

Objek CLOB memiliki dukungan penuh terhadap transaksional. Perubahan yang dilakukan melalui SQL, *package* DBMS_LOB, atau *Oracle Call Interface* (OCI) berpartisipasi penuh dalam transaksi. Manipulasi nilai CLOB dapat dilakukan dan dibatalkan. Namun, Anda tidak dapat menyimpan locator CLOB dalam variabel PL / SQL atau OCI dalam satu transaksi dan kemudian menggunakannya dalam transaksi atau *session* lain.

NCLOB
Tipe data NCLOB menyimpan data Unicode. Didukung oleh set karakter lebar-tetap dan variabel-lebar, dan menggunakan karakter nasional yang telah ditetapkan. Objek NCLOB dapat menyimpan sampai dengan (4 gigabyte -1) * (nilai parameter CHUNK LOB *storage*) dari data karakter teks. Jika tablespace dalam database Anda memiliki ukuran blok standar, dan jika Anda telah menggunakan nilai default parameter chunk LOB *storage* saat membuat kolom LOB, maka ini setara dengan (4 gigabyte - 1) * (*database block size*) .

Objek NCLOB memiliki dukungan penuh terhadap transaksional. Perubahan yang dilakukan melalui SQL, *package* DBMS_LOB, atau OCI berpartisipasi penuh dalam transaksi. Manipulasi nilai NCLOB dapat dilakukan dan dibatalkan. Namun, Anda tidak dapat menyimpan *locator* NCLOB dalam variabel PL / SQL atau OCI dalam satu transaksi dan kemudian menggunakannya dalam transaksi transaksi atau *session* lain.

Tipe Data Rowid
Tipe data Rowid terdiri dari tipe data ROWID dan tipe data UROWID.

ROWID
Baris-baris dalam tabel yang terorganisir dalam Oracle Database memiliki alamat baris yang disebut *rowids*. Anda dapat memeriksa alamat baris rowid oleh *query pseudocolumn* ROWID.

Nilai dari *pseudocolumn* ini adalah string yang mewakili alamat dari tiap baris. String ini memiliki tipe data ROWID. Anda juga dapat membuat tabel dan cluster yang berisi kolom yang memiliki tipe data ROWID. Oracle Database tidak menjamin bahwa nilai kolom tersebut adalah rowids yang valid.

Catatan:
Dimulai pada Oracle8, Oracle SQL mengembangkan format rowids untuk mendukung efisiensi tabel partisi dan indeks serta alamat data blok tablespace- relatif tanpa ambiguitas. Jika Anda menjalankan Versi 7 dari database dan Anda berniat untuk melakukan upgrade, gunakan *package* DBMS_ROWID untuk bermigrasi dalam data Anda ke format rowids *extended*.

Rowids berisi informasi berikut:

- Data blok dari datafile yang mengandung baris. Panjang string ini tergantung pada sistem operasi Anda.
- Baris di blok data.
- File database yang berisi baris. Datafile yang pertama memiliki nomor 1. Panjang string tergantung pada sistem operasi Anda.
- Data jumlah objek, yang merupakan nomor identifikasi ditugaskan untuk setiap segmen database. Anda dapat mengambil objek data nomor dari *view* USER_OBJECTS, DBA_OBJECTS, dan ALL_OBJECTS. Objek yang memiliki segmen yang sama (*clustered* tabel di cluster yang sama, misalnya) memiliki nomor objek yang sama.

Rowids disimpan sebagai basis 64 nilai yang dapat berisi karakter AZ, az, 0-9, dan tanda plus (+) dan garis miring (/). Rowids tidak tersedia secara langsung. Anda dapat menggunakan *package* yang disediakan DBMS_ROWID untuk menafsirkan isi rowid. Fungsi *package* adalah melakukan ekstrak dan memberikan informasi mengenai empat unsur rowid yang tercantum di atas.

UROWID
Baris-baris dari beberapa tabel memiliki alamat yang bukan fisik atau permanen yang tidak di*generate* oleh Oracle Database. Misalnya, alamat baris dari tabel-tabel terindeks-terorganisir disimpan dalam indeks, yang dapat dipindahkan (disimpan di tempat lain). Rowids tabel asing (seperti tabel DB2 yang diakses melalui gateway) tidak memiliki standar Oracle rowids.

Oracle menggunakan universal rowids (urowids) untuk menyimpan alamat dari tabel terindeks-terorganisir dan asing. Indeks tabel yang terorganisir memiliki urowids logik dan tabel asing telah urowids asing. Kedua jenis urowid disimpan dalam pseudocolumn ROWID (sebagai rowids fisik tabel tumpukan-terorganisir).

Oracle menciptakan rowids logis berdasarkan *primary key* tabel. Rowids logis tidak berubah selama kunci primer tidak berubah. Pseudocolumn ROWID dari tabel indeks-terorganisir memiliki tipe data UROWID. Anda dapat mengakses pseudocolumn ini seperti yang akan Anda lakukan pada pseudocolumn ROWID tabel tumpukan-terorganisir (menggunakan statemen SELECT ... ROWID). Jika Anda ingin menyimpan rowids sebuah tabel indeks-terorganisir, maka anda dapat menentukan tipe kolom UROWID untuk tabel tersebut dan mengambil nilai pseudocolumn ROWID ke dalam kolom tersebut.

Tipe Data Karakter

Tipe Data	Oracle 9i	Oracle 10g	Oracle 11g	Keterangan
char(size)	Maksimum *size* 2000 byte.	Maksimum *size* 2000 byte.	Maksimum *size* 2000 byte.	Dimana *size* adalah jumlah karakter untuk menyimpan. *Fixed-length strings*. Spasi dihitung

nchar(size)	Maksimum *size* 2000 byte.	Maksimum *size* 2000 byte.	Maksimum *size* 2000 byte.	Dimana size adalah jumlah karakter untuk menyimpan. *Fixed-length NLS string*. Spasi dihitung.
nvarchar2(size)	Maksimum *size* 4000 byte.	Maksimum *size* 4000 byte.	Maksimum *size* 4000 byte.	Dimana size adalah jumlah karakter untuk menyimpan. *Variable-length NLS string*.
varchar2(size)	Maksimum *size* 4000 byte.	Maksimum *size* 4000 byte.	Maksimum *size* 4000 byte.	Dimana size adalah jumlah karakter untuk menyimpan. *Variable-length NLS string*.
long	Maksimum *size* 2 GB.	Maksimum *size* 2 GB.	Maksimum *size* 2 GB.	*Variable-length strings. (backward compatible)*
raw	Maksimum size 2 GB.	Maksimum size 2 GB.	Maksimum size 2 GB.	*Variable-length binary strings*
long raw	Maksimum *size* 2 GB.	Maksimum *size* 2 GB.	Maksimum *size* 2 GB.	*Variable-length binary strings. (backward compatible)*

Tipe Data Numerik

Tipe Data	Oracle 9i	Oracle 10g	Oracle 11g	Keterangan
number(p,s)	Presisi dapat berupa nilai antara 1 s/d 38. Skala dapat berupa nilai antara -84 s/d 127.	Presisi dapat berupa nilai antara 1 s/d 38. Skala dapat berupa nilai antara -84 s/d 127.	Presisi dapat berupa nilai antara 1 s/d 38. Skala dapat berupa nilai antara -84 s/d 127.	Dimana p adalah presisi, s adalah skala. Misalnya number (7,2) artinya nilai nomor dengan 5 digit sebelum desimal, dan 2 digit setelah desimal.
float				
dec(p,s)	Presisi dapat berupa nilai antara 1 s/d 38.	Presisi dapat berupa nilai antara 1 s/d 38.	Presisi dapat berupa nilai antara 1 s/d 38.	Dimana p adalah presisi, s adalah skala. Misalnya number (7,2) artinya nilai nomor dengan 5 digit sebelum desimal, dan 2 digit setelah desimal.
decimal(p,s)	Presisi dapat berupa nilai antara 1 s/d 38.	Presisi dapat berupa nilai antara 1 s/d 38.	Presisi dapat berupa nilai antara 1 s/d 38.	Dimana p adalah presisi, s adalah skala. Misalnya number (7,2) artinya nilai nomor dengan 5 digit sebelum desimal, dan 2 digit setelah desimal.
integer				
int				
smallint				
real				
double precision				

Tipe Data Tanggal / Waktu

Tipe Data	Oracle 9i	Oracle 10g	Oracle 11g	Keterangan
date	Tanggal antara 11 Januari 4712 SM s/d 31 Desember 9999 M.	Tanggal antara 11 Januari 4712 SM s/d 31 Desember 9999 M.	Tanggal antara 11 Januari 4712 SM s/d 31 Desember 9999 M.	
timestamp	*fractional seconds precision* harus	*fractional seconds precision* harus	*fractional seconds precision* harus	Terdiri dari Tahun, Bulan, Hari, Jam,

(fractional seconds precision)	berisi nomor antara 0 s/d 9 (dengan nilai default 6)	berisi nomor antara 0 s/d 9 (dengan nilai default 6)	berisi nomor antara 0 s/d 9 (dengan nilai default 6)	Menit dan Detik. Contoh : timestamp(6)
timestamp (fractional seconds precision) with time zone	*fractional seconds precision* harus berisi nomor antara 0 s/d 9 (dengan nilai default 6)	*fractional seconds precision* harus berisi nomor antara 0 s/d 9 (dengan nilai default 6)	*fractional seconds precision* harus berisi nomor antara 0 s/d 9 (dengan nilai default 6)	Terdiri dari Tahun, Bulan, Hari, Jam, Menit dan Detik. ; dengan nilai zona waktu yang telah ditetapkan. Contoh : timestamp(5) with time zone
timestamp (*fractional seconds precision*) with local time zone	*fractional seconds precision* harus berisi nomor antara 0 s/d 9 (dengan nilai default 6)	*fractional seconds precision* harus berisi nomor antara 0 s/d 9 (dengan nilai default 6)	*fractional seconds precision* harus berisi nomor antara 0 s/d 9 (dengan nilai default 6)	Terdiri dari Tahun, Bulan, Hari, Jam, Menit dan Detik. ; dengan nilai zona waktu yang telah ditetapkan pada sesi lokal. Contoh : timestamp(4) with time zone
interval year (*year precision*) to month	*year precision* adalah digit nomor dalam tahun. (dengan nilai default 2)	*year precision* adalah digit nomor dalam tahun. (dengan nilai default 2)	*year precision* adalah digit nomor dalam tahun. (dengan nilai default 2)	Periode watu disimpan dalam tahun dan bulan. Contoh : interval year(4) to month
interval day (day precision) to second (fractional seconds precision)	*day precision* berisi nomor antara 0 s/d 9 (dengan nilai default 2) *fractional seconds precision* berisi nomor antara 0 s/d 9 (dengan nilai default 6)	*day precision* berisi nomor antara 0 s/d 9 (dengan nilai default 2) *fractional seconds precision* berisi nomor antara 0 s/d 9 (dengan nilai default 6)	*day precision* berisi nomor antara 0 s/d 9 (dengan nilai default 2) *fractional seconds precision* berisi nomor antara 0 s/d 9 (dengan nilai default 6)	Periode watu disimpan dalam hari, jam, menit dan detik. Contoh : interval day(2) to second(6)

Tipe Data Large Object (LOB)

Tipe Data	Oracle 9i	Oracle 10g	Oracle 11g	Keterangan
bfile	Maksimum *size* 4 GB.	Maksimum file *size* 2^{32}*-1 bytes*	Maksimum file *size* 2^{64}*-1 bytes*	File *locators* yang akan menunjukkan binary file ke *filesystem* server. (diluar database)
blob	Disimpan dalam lebih dari 4 GB data binary	Disimpan dalam lebih dari (4 gigabytes -1) * (nilai CHUNK parameter dari LOB storage).	Disimpan dalam lebih dari (4 gigabytes -1) * (nilai CHUNK parameter dari LOB storage).	Disimpan dalam unstructured binary large objects
clob	Disimpan dalam lebih dari 4 GB data karakter	Disimpan dalam lebih dari (4 gigabytes -1) * (nilai CHUNK parameter dari LOB storage) data karakter.	Disimpan dalam lebih dari (4 gigabytes -1) * (nilai CHUNK parameter dari LOB storage) data karakter.	Disimpan dalam *single-byte* dan *multi-byte character data.*
nclob	Disimpan dalam lebih dari 4 GB data karakter teks	Disimpan dalam lebih dari (4 gigabytes -1) * (nilai CHUNK parameter dari LOB storage) data karakter teks.	Disimpan dalam lebih dari (4 gigabytes -1) * (nilai CHUNK parameter dari LOB storage) data karakter teks.	Disimpan dalam *unicode data*

Tipe Data Rowid

Tipe Data	Oracle 9i	Oracle 10g	Oracle 11g	Keterangan
rowid	Format dari rowid adalah : BBBBBBB.RRRR.FFFFF Dimana BBBBBBB adalah block dalam database file; RRRR adalah row dalam block; FFFFF adalah database file.	Format dari rowid adalah : BBBBBBB.RRRR.FFFFF Dimana BBBBBBB adalah block dalam database file; RRRR adalah row dalam block; FFFFF adalah database file.	Format dari rowid adalah : BBBBBBB.RRRR.FFFFF Dimana BBBBBBB adalah block dalam database file; RRRR adalah row dalam block; FFFFF adalah database file.	*Fixed-length binary data*. Setiap record dalam database mempunyai sebuah physical address atau **rowid**.
urowid				Universal rowid. DImana ***size*** bersifat opsional.

Suatu jenis data baru di Oracle 11g: SIMPLE_INTEGER

Sebelum ke Oracle 11g, kita telah menggunakan tipe data PLS_INTEGER dalam program PL / SQL. Dalam 11g, sebuah tipe data baru, SIMPLE_INTEGER, telah diperkenalkan. Ini adalah sub-jenis tipe data PLS_INTEGER dan memiliki rentang yang sama dengan PLS_INTEGER. Perbedaan dasar antara keduanya adalah SIMPLE_INTEGER yang selalu NOT NULL. Ketika nilai dari variabel dinyatakan tidak pernah akan null maka kita dapat mendeklarasikan dengan tipe data SIMPLE_INTEGER. Perbedaan utamanya adalah bahwa ia pernah memberikan pesan *error* overflow seperti induknya, tipe data numerik melainkan tetap menjalankannya tanpa memberikan pesan *error*. Ketika kita tidak perlu khawatir memeriksa status null serta melimpahkan kesalahan, tipe data SIMPLE_INTEGER adalah yang terbaik untuk digunakan.

Perbedaan lainnya adalah bahwa tipe data SIMPLE_INTEGER memberikan peningkatan kinerja yang besar dibandingkan PLS_INTEGER ketika kode ini disusun dalam mode 'NATIVE', karena operasi aritmatika pada tipe SIMPLE_INTEGER dilakukan langsung di tingkat hardware. Ketika kode dikompilasi dalam mode 'diartikan' yang merupakan default, SIMPLE_INTEGER masih lebih baik daripada PLS_INTEGER tetapi tidak begitu signifikan.

Anda dapat mengubah sesi untuk mengkompilasi kode dalam mode 'NATIVE' dengan menjalankan pernyataan berikut ini dan kemudian kompilasi prosedur yang tersimpan.

SQL> ALTER SESSION SET PLSQL_CODE_TYPE 'NATIVE' =;

Session Altered.

Untuk beralih kembali ke modus standar,

SQL> ALTER SESSION SET PLSQL_CODE_TYPE = 'diartikan';

Session Altered.

Potongan kode Setelah menunjukkan kesalahan ketika variabel SIMPLE_INTEGER diberi nilai null.

DECLARE

```
v_col1 SIMPLE_INTEGER: = 1;
BEGIN
v_col1: = NULL;
END;
/
```

Sejak v_col1 harus selalu bukan null dan kita berusaha untuk memberikan nilai null, memberikan kesalahan berikut.

PLS-00382: Expression is of wrong type

Bahkan jika kita tidak memberi nilai dalam bagian deklarasi untuk tipe data SIMPLE_INTEGER, itu menghasilkan menjadi kesalahan berikut:

PLS-00218: a variable declared NOT NULL must have an initialization assignment

Penutup

Setiap Tipe Data dalam Oracle Database 11g memiliki karakteristik serta perlakuan yang berbeda-beda. Dengan mengetahui masing-masing karakteristik dari Tipe-tipe Data tersebut, diharapkan seorang Database Administrator (DBA) dapat memaksimalkan kinerja database yang mereka rancang, yang salah satu faktor pendukungnya adalah pemakaian tipe data yang sesuai pada setiap tabel yang digunakan dalam database. Tujuan lain dari penyusunan tulisan ini adalah agar setiap DBA dapat menggunakan Resource Database yang tersedia secara efisien, karena telah menggunakan tipe data yang sesuai dengan nilai ataupun informasi yang mereka butuhkan dalam database masing-masing.

Referensi

Oracle/PLSQL: Data Types
http://www.techonthenet.com/oracle/datatypes.php
PL/SQL Datatypes
http://download.oracle.com/docs/cd/B19306_01/appdev.102/b14261/datatypes.htm
Datatypes in Oracle
http://www.go4expert.com/forums/showthread.php?t=4210
A new data type in Oracle 11g: SIMPLE_INTEGER
http://decipherinfosys.wordpress.com/2008/03/13/a-new-data-type-in-oracle-11g-simple_integer/

Biografi Penulis

Yuafanda Kholfi Hartono. Menyelesaikan S1 di Universitas Indonesia. Sebelumnya menamatkan pendidikan di Sekolah Tinggi Akuntansi Negara, minat dan ketertarikan pada bidang IT membuat pada akhirnya ditempatkan pada Direktorat Jenderal Bea dan Cukai Kementerian Keuangan sebagai Database Administrator (DBA).