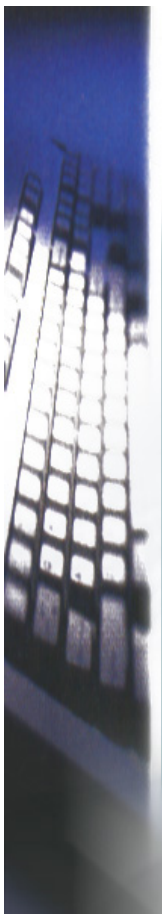
Constraint dan Manajemen Data dalam Timezone Berbeda

Rosa Ariani Sukamto
Email: rosa_if_itb_01@yahoo.com
Blog: http://udinrosa.wordpress.com
Website: http://www.gangsir.com



Constraint

- Aturan pada basis data
- Dapat mencegah penghapusan tabel jika ada tabel yang bergantung dengan tabel lain dihapus
- Dapat mengatur proses **`insert, update, delete`** data
- Sebaiknya diberi nama sendiri agar mudah diingat, karena nama standar yang diberikan Oracle mungkin tidak mudah diingat

Tipe-tipe Constraint

- **PRIMARY KEY** (pk)
  - mendefinisikan kunci primer *(primary key)* dari sebuah atau kumpulan *field* yang digunakan sebagai kunci primer
  - bersifat unik dan tidak boleh bernilai NULL
- **FOREIGN KEY** (fk)
  - mendefinisikan kunci luar *(foreign key)* dari sebuah atau kumpulan *field* yang digunakan sebagai kunci luar untuk relasi
  - untuk menjaga relasi/integritas dapat menggunakan
    - **ON DELETE CASCADE**
      - menghapus beserta data yang ada untuk koneksi kunci luar
    - **ON DELETE SET NULL**
      - jika dihapus maka data relasi kunci luar akan diset NULL.

Tipe-tipe Constraint

- **UNIQUE** (uk)
  - mengeset sebuah *field* menjadi unik (tidak boleh ada yang sama)
- **COMPOSITE UNIQUE KEY** (uk)
  - mengeset kumpulan *field* sebagai kombinasi bernilai unik
  - hanya bisa digunakan untuk obyek tabel
- **CHECK** (ck)
  - untuk mengecek constraint sesuai dengan kondisi cek yang diinginkan
- **NOT NULL** (nn)
  - tidak membolehkan sebuah *field* bernilai NULL
  - hanya dapat didefinisikan untuk kolom

Level Constraint

- Column level
  - Constraint yang dapat digunakan untuk kolom pada tabel
  - Hanya berpengaruh untuk kolom tabel
- Table level
  - Constraint yang dapat digunakan untuk tabel dan keperluan tabel
  - Hanya berpengaruh untuk tabel

Query Constraint

CONSTRAINT <*constraint name*>
<***TYPE OF CONSTRAINT***> **(**<*column name or condition*>**)**
CONSTRAINT <*constraint name*>
PRIMARY KEY (<*column 1*>**,** <*column 2*>**,** <*column n*>**)**
CONSTRAINT <*constraint name*>
FOREIGN KEY (<*column*>**)**
REFERENCES <*referenced table*> **(**<*referenced primary or unique key column*>**)**
[**ON DELETE CASCADE/SET NULL**]
CONSTRAINT <*name*> **UNIQUE (**<*column*>**),**
CONSTRAINT <*name*> **UNIQUE (**<*column 1*>**,** <*column 2*>**,** <*column n*>**)**
CONSTRAINT <*name*> **CHECK (**<*conditional expression*>**)**

Query Constraint

```
CREATE TABLE err_test (
  widget_name VARCHAR2(100),
  widget_count NUMBER ,
  CONSTRAINT no_small_numbers
  CHECK
  (widget_count > 1000));
```

Query Alter Tabel Constraint

ALTER TABLE <*table*>
ADD CONSTRAINT <constraint name>
<***TYPE OF CONSTRAINT***> **(**<*column*>**)**
ADD CONSTRAINT <constraint name>
<***TYPE OF CONSTRAINT***> **(**<*column*>**)**
REFERENCES <*table*>**(**<*primary key column*>**)**
[**ON DELETE CASCADE/NULL**]
MODIFY (<*column*>
CONSTRAINT <*constraint name*> **NOT NULL)**
DROP <***TYPE OF CONSTRAINT***> **(**<*column*>**)**
CONSTRAINT <*constraint name*> [**CASCADE**]
DROP PRIMARY KEY CASCADE
DISABLE CONSTRAINT <*constraint name*> [**CASCADE**]
ENABLE CONSTRAINT <*constraint name*>

Melihat Constraint

```
SELECT <constraint name>,
<TYPE OF CONSTRAINT>
FROM USER_CONSTRAINTS
WHERE TABLE_NAME = '<table>'
```

Tipe Data untuk Menangani Waktu: Datetime dan Tipe Interval

- Tipe-tipe pada bagian ini mengijinkan kita untuk menyimpan dan memanipulasi tanggal, waktu, dan interval (periode waktu).
- Variabel yang memiliki tipe data tanggal/waktu menyimpan nilai-nilai yang disebut datetimes;
- variabel yang memiliki tipe-tipe data interval menyimpan nilai-nilai yang disebut dengan interval.
- Datetime atau interval terdiri dari *field-field*, yang menentukan nilai-nilainya.

Nama Field	Nilai Datetime Valid	Nilai Interval Valid
YEAR	-4712 hingga 9999 (tidak termasuk tahun 0)	Setiap integer bukan nol
MONTH	0 hingga 12	0 hingga 11
DAY	01 hingga 31 (dibatasi oleh nilai MONTH dan YEAR, berdasarkan aturan kalender di tempat terjadinya peristiwa)	Setiap integer bukan nol
HOUR	00 hingga 21	0 hingga 23
MINUTE	00 hingga 59	0 hingga 59
SECOND	00 hingga 59.9(n), dimana 9(n) merupakan presisi dari detik waktu fraksional	0 hingga 59.9(n), dimana 9(n) merupakan presisi dari detik interval fraksional
TIMEZONE_HOUR	-12 hingga 14 (jangkauan mengakomodasi perubahan waktu siang hari)	Tidak dapat digunakan
TIMEZONE_MINUTE	00 hingga 59	Tidak dapat digunakan
TIMEZONE_REGION	Dapat ditemukan pada V$TIMESONZE_NAMES	Tidak dapat digunakan
TIMEZONE_ABBR	Dapat ditemukan pada V$TIMESONZE_NAMES	Tidak dapat digunakan

Datetime dan Tipe Interval

- Kecuali TIMESTAMP WITH LOCAL TIMEZONE, tipe-tipe ini seluruhnya merupakan bagian dari standar SQL92.

Date

- Menyimpan tanggal dan waktu
- **SYSDATE** menghasilkan tanggal dan waktu saat ini.
- Tanggal-tanggal yang valid berkisar antara 1 Januari 4712 BC hingga 31 Desember 9999 AD (Julian Date = 'J').
- Kita dapat menggunakan model format tanggal 'J' dengan function **TO_DATE** dan **TO_CHAR** untuk mengkonversi antara nilai-nilai **DATE** dan nilai persamaan Julian-nya.

.....

```
order_date BETWEEN
TO_DATE('01-Jan-03') AND
TO_DATE('31-Jan-03');
```

Date

- Kita dapat menambah dan mengurangi tanggal. Contohnya, perintah berikut ini menghasilkan jumlah hari sejak seorang karyawan diperkerjakan:

```
SELECT SYSDATE - hiredate
INTO days_worked
FROM emp
WHERE empno = 7499;
```

Dalam ekspresi-ekspresi aritmatika, PL/SQL menginterpretasikan literal-literal integer sebagai hari-hari. Sebagai contoh, SYSDATE+1 adalah besok.

Timestamp

- Tipe date TIMESTAMP, yang merupakan perluasan dari tipe data DATE, menyimpan tahun, bulan, hari, menit, dan detik. Sintaksnya:

```
TIMESTAMP[(precision)]
```

Pada contoh berikut ini, kita mendeklarasikan variable bertipe TIMESTAMP, lalu memberikan nilai literal terhadapnya:

```
DECLARE checkout TIMESTAMP(3);
BEGIN checkout := '1999-06-22
07:48:53.275';
 ...
END;
```

Dalam contoh ini, bagian kecil dari field detik adalah 0.275.

Timestamp with Time Zone

- Tipe data TIMESTAMP WITH TIMEZONE, yang memperluas tipe data TIMESTAMP, dengan menambahkan *time-zone displacement*.
- Perbedaan time-zone merupakan perbedaan (dalam jam dan menit) antara waktu local dengan Coordinated Universal Time (UTC) – dahulu Greenwich Mean Time. Sintaksnya adalah:

```
TIMESTAMP[( precision)] WITH TIME ZONE
```

dimana parameter opsional *precision* menentukan jumlah digit di dalam sebagian kecil field detik. Kita tidak dapat menggunakan konstanta simbolik atau variable untuk menentukan *precision*; kita harus menggunakan literal integer dalam jangkauan 0..9. Default-nya adalah 6.

Timestamp with Time Zone

- Pada contoh berikut ini, kita mendeklarasikan variable bertipe TIMESTAMP WITH TIME ZONE, lalu memberikan nilai literal terhadapnya:

```
DECLARE logoff TIMESTAMP(3) WITH
TIME ZONE;
BEGIN
   logoff := '1999-10-31
09:42:37.114 + 02:00';
   ...
END;
```

Pada contoh ini, perbedaan time-zone adalah +02:00.

Timestamp with Local Time Zone

- Tipe data TIMESTAMP WITH LOCAL TIME ZONE, yang memperluas tipe data TIMESTAMP, dengan menambahkan *time-zone displacement*.
- Perbedaan time-zone merupakan perbedaan (dalam jam dan menit) antara waktu local dengan Coordinated Universal Time (UTC) – dahulu Greenwich Mean Time.
- Kita juga dapat menggunakan time zone bernama, seperti dengan TIMESTAMP WITH TIME ZONE. Sintaksnya adalah:

```
TIMESTAMP[( precision)]
WITH LOCAL TIME ZONE
```

dimana parameter opsional *precision* menentukan jumlah digit di dalam sebagian kecil field detik. Kita tidak dapat menggunakan konstanta simbolik atau variable untuk menentukan *precision*; kita harus menggunakan literal integer dalam jangkauan 0..9. Default-nya adalah 6.

Timestamp with Local Time Zone

- Pada contoh berikut ini, kita mendeklarasikan variable bertipe TIMESTAMP WITH TIME ZONE:

```
DECLARE logoff TIMESTAMP(3) WITH
LOCAL TIME ZONE;
BEGIN
   ...
END;
```

Kita tidak dapat memberikan nilai-nilai literal kepada variable dengan tipe ini.

Interval Year to Month

- Kita menggunakan tipe data INTERVAL YEAR TO MONTH untuk menyimpan dan memanipulasi interval-interval dari tahun dan bulan. Sintaksnya:

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

dimana parameter opsional *precision* menentukan jumlah digit di dalam field tahun. Kita tidak dapat menggunakan konstanta simbolik atau variable untuk menentukan *precision*; kita harus menggunakan literal integer dalam jangkauan 0..4. Default-nya adalah 2.

Interval Year to Month

- Dalam contoh berikut ini, kita mendeklarasikan variable bertipe INTERVAL YEAR TO MONTH, lalu memberikan nilai 101 tahun dan 3 bulan kepadanya:

```
DECLARE lifetime INTERVAL YEAR(3) TO MONTH;
BEGIN
    lifetime := INTERVAL '101-3'
YEAR TO MONTH;
    lifetime := INTERVAL '101' YEAR;
    lifetime := INTERVAL '3' MONTH;

     ...
 END;
```

Interval Day to Second

- Kita menggunakan tipe data INTERVAL DAY TO SECOND untuk menyimpan dan memanipulasi interval-interval dari hari, jam, menit, dan detik. Sintaksnya adalah:

```
INTERVAL DAY[( leading_precision)]
TO SECOND[( fractional_seconds_precision)]
```

dimana *leading_precision* dan *fractional_seconds_precision* menentukan jumlah digit dalam field hari dan detik. Dalam kedua kasus ini, kita tidak dapat menggunakan konstanta simbolik atau variable untuk menentukan precision; kita harus menggunakan literal integer dalam jangkauan 0..9. Defaultnya adalah 2 dan 6.

Interval Day to Second

- Dalam contoh berikut, kita mendeklarasikan tipe INTERVAL DAY TO SECOND:

```
DECLARE lag_time INTERVAL
DAY(3) TO SECOND(3);
BEGIN
   IF lag_time > INTERVAL '6'
DAY
   THEN ... ...
END;
```

Contoh Query

```
create table test(
tanggal TIMESTAMP(3),
logoff TIMESTAMP(3) WITH
  TIME ZONE,
login TIMESTAMP(3) WITH
  LOCAL TIME ZONE,
lifetime INTERVAL YEAR(3) TO
  MONTH);
```