**ADVANCE SQL**

**DATE FUNCTION**

Fungsi yang digunakan untuk melakukan manipulasi *date*(tanggal) dan *time*(waktu) pada database.

* **CURRENT\_DATE :** Menampilkan tanggal hari ini.

**Syntax:** SELECT CURRENT\_DATE();

**Output:**2022-01-16

* **CURRENT\_TIME :** Menampilkan waktu saat ini.

**Syntax:** SELECT CURRENT\_TIME();

**Output:**02:53:15

* **CURRENT\_TIMESTAMP :** Menampilkan tanggal dan waktu saat ini.

**Syntax:** SELECT CURRENT\_TIMESTAMP();

**Output:**2022-01-10 02:53:21

* **EXTRACT :** Menampilkan kembali satu bagian dari tanggal / waktu.

**Syntax:** SELECT EXTRACT(part FROM date)

SELECT EXTRACT(MONTH FROM "2018-07-16");

**Output:**7

SELECT EXTRACT(HOUR FROM CURRENT\_TIMESTAMP);

**Output:**3

SELECT EXTRACT(YEAR FROM CURRENT\_TIMESTAMP);

**Output:**2022

SELECT EXTRACT(WEEK FROM CURRENT\_TIMESTAMP);

**Output:**4

* **DATE\_TRUNC :** Memotong ke presisi yang ditentukan.

**Syntax:** SELECT DATE\_TRUNC(‘[interval]’, time\_column)

SELECT DATE TRUNC (‘MONTH’, CURRENT\_TIMESTAMP);

**Output:**2022-01-01 00:00:00

SELECT DATE TRUNC (‘QUARTER’, CURRENT\_TIMESTAMP);

**Output:**2022-01-01 00:00:00

**KEY**

**Key**atau kunci adalah satu atau gabungan dari beberapa atribut yang dapat membedakan semua baris pada data dalam suatu tabel secara unik, artinya jika suatu atribut dijadikan sebuah key maka tidak boleh ada baris data yang mempunyai nilai yang saman untuk atribut tersebut.

* **Super Key**

Satu atau lebih atribut yang dapat membedakan setiap baris data dalam sebuah tabel secara unik. Bisa jadi ada lebih dari satu kumpulan atribut yang yang bersifat super key dalam sebuah tabel.

* **Candidate Key**

Merupakan kumpulan atribut minimal yang dapat membedakan setiap baris dalam sebuah tabel secara unik. Sebuah candidate key tidak boleh berisi atribut atau kumpulan atribut yang telah menjadi superkey yang lain.

* **Primary Key**

Primary Key adalah atribut candidate key yang telah dipilih untuk mengidentifikasi setiap record secara unik. Primary Key harus merupakan atribut yang benar-benar unik dan tidak boleh ada nilai NULL.

* **Composite Key**

Composite Key adalah key yang terdiri dari dua atau lebih atribut yang secara unik mengidentifikasi suatu entitas. Composite key terjadi karena dalam sebuah tabel tidak ditemukannya sebuah primary key, jadi agar tidak melakukan pembuatan primary key baru maka dibuatlah sebuah composite key yaitu gabungan dari dua buah foreign key, sehingga tidak ada data yang sama yang akan dimunculkan.

* **Foreign Key**

Jika sebuah primary key terhubung ke tabel lain, maka keberadaan primary key disebut sebagai foreign key. Foreign key adalah atribut dalam satu relasi yang digunakan untuk menunjuk ke suatu baris pada relasi yang lain, jadi foreign key digunakan untuk membuat sebuah relasi yang terjadi antara tabel A dengan tabel B, dimana ketika tabel A membuat sebuah relasi dengan tabel B maka di tabel B primary key tabel A akan menjadi foreign key di tabel B.

* **Secondary/Alternative Key**

Alternative Key adalah primary key yang tidak terpilih. Misalnya dalam suatu entitas terdapat dua atribut yang bisa dijadikan sebagai primary key. Sementara yang boleh dijadikan primary hanya satu, maka kita harus memilih salah satu. Atribut yang dipilih, disebut primary key. Sedangkan atribut yang tidak dipilih disebut dengan alternative key.

* **Non-Key Attributes**

Atribut selain dari candidate key pada tabel.

* **Non-Prime Attribute**

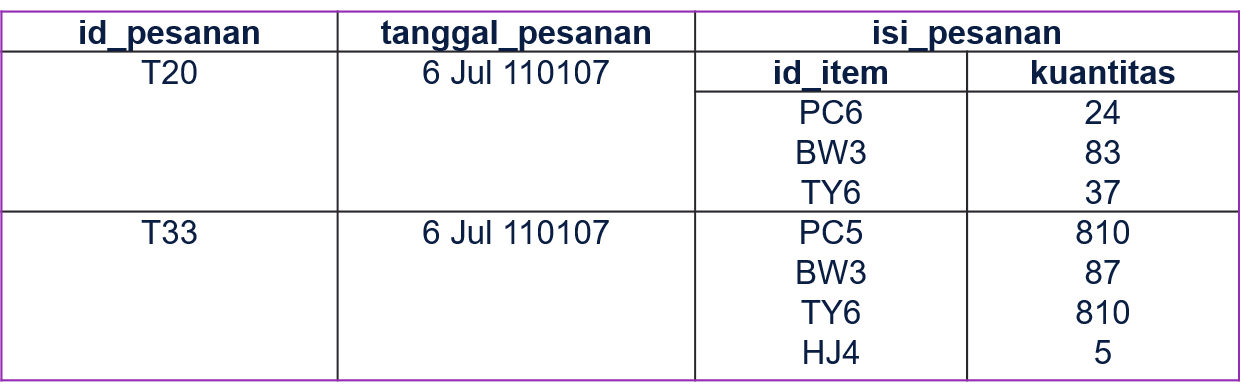
Atribut selain dari primary key pada tabel.

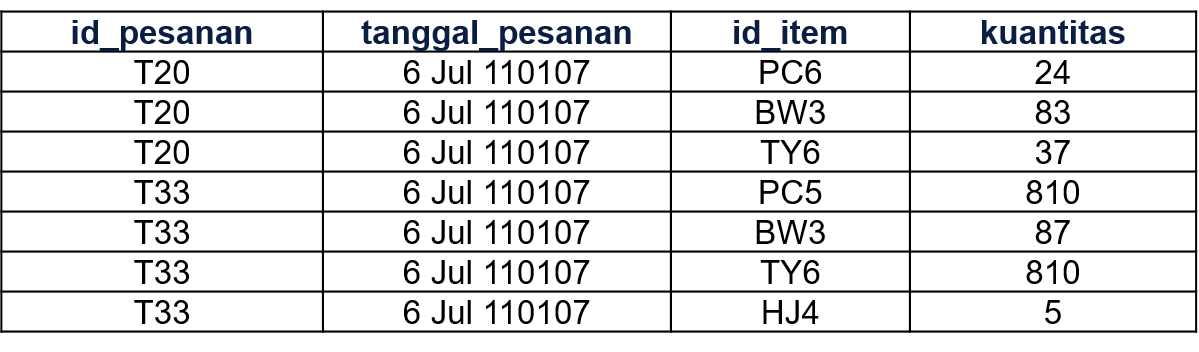
**DATABASE NORMALIZATION**

* **First Normal Form (1NF)**

Suatu relasi disebut memenuhi bentuk **normal** pertama (**1NF**) jika dan hanya jika setiap atribut dari relasi tersebut hanya memiliki nilai tunggal dan tidak ada pengulangan grup atribut dalam baris. Bentuk normal yang pertama atau 1NF mensyaratkan beberapa kondisi dalam sebuah database, berikut adalah fungsi dari bentuk normal pertama ini :

* Menghilangkan duplikasi kolom dari tabel yang sama.
* Buat tabel terpisah untuk masing-masing kelompok data terkait dan mengidentifikasi setiap baris dengan kolom yang unik (primary key).





* **Second Normal Form (2NF)**

Syarat untuk menerapkan normalisasi bentuk kedua ini adalah data telah dibentuk dalam 1NF, berikut adalah beberapa fungsi normalisasi 2NF :

* Menghapus beberapa subset data yang ada pada tabel dan menempatkan mereka pada tabel terpisah.
* Menciptakan hubungan antara tabel baru dan tabel lama dengan menciptakan foreign key.
* Tidak ada atribut dalam tabel yang secara fungsional bergantung pada candidate key tabel tersebut.

|  |  |  |  |
| --- | --- | --- | --- |
| **id\_pekerja** | **id\_proyek** | **dana\_proyek** | **total\_waktu** |
| P001 | Pj001 | 20 | 20 |
| P003 | Pj001 | 20 | 16 |
| P002 | Pj002 | 17 | 35 |
| P002 | Pj003 | 84 | 42 |
| P003 | Pj002 | 17 | 17 |
| P004 | Pj001 | 20 | 83 |
| P004 | Pj003 | 84 | 41 |
| - | Pj004 | 100 | - |

|  |  |
| --- | --- |
| **id\_poyek** | **dana\_proyek** |
| pj001 | 20 |
| pj002 | 17 |
| pj003 | 84 |
| pj004 | 100 |

|  |  |  |
| --- | --- | --- |
| **id\_pekerja** | **id\_proyek** | **total\_waktu** |
| P001 | Pj001 | 20 |
| P003 | Pj001 | 16 |
| P002 | Pj002 | 35 |
| P002 | Pj003 | 42 |
| P003 | Pj002 | 17 |
| P004 | Pj001 | 83 |
| P004 | Pj003 | 41 |

* **Third Normal Form (3NF)**

Normalisasi database dalam bentuk 3NF bertujuan untuk menghilangkan seluruh atribut atau field yang tidak berhubungan dengan primary key. Dengan demikian tidak ada ketergantungan transitif pada setiap kandidat key.

Syarat dari bentuk normal ketiga atau 3NF adalah :

* Memenuhi semua persyaratan dari bentuk normal kedua.
* Menghapus kolom yang tidak tergantung pada primary key.
* Setiap atribut yang bukan kunci tidak tergantung secara fungsional terhadap atribut bukan kunci yang lain dalam relasi tsb (tidak terdapat ketergantungan transitif pada atribut bukan kunci).

|  |  |  |
| --- | --- | --- |
| **id\_proyek** | **manajer** | **tanggal\_lahir** |
| pj001 | Jon | 24 Jan 11065 |
| pj002 | Viki | 23 Mar 11070 |
| pj003 | Jon | 24 Jan 11065 |
| pj004 | Marini | 06 Jul 11060 |

|  |  |
| --- | --- |
| **id\_proyek** | **manajer** |
| pj001 | Jon |
| pj002 | Viki |
| pj003 | Jon |
| pj004 | Marini |

|  |  |
| --- | --- |
| **manajer** | **tanggal\_lahir** |
| Jon | 24 Jan 11065 |
| Viki | 23 Mar 11070 |
| Marini | 6 Jul 11060 |

**JOIN & MERGE**

Join adalah penggabungan table yang dilakukan melalui kolom / key tertentu yang memiliki nilai terkait untuk mendapatkan satu set data dengan informasi lengkap.

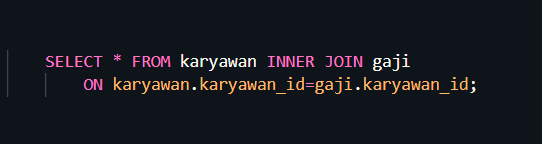
INNER JOIN

LEFT JOIN

RIGHT JOIN

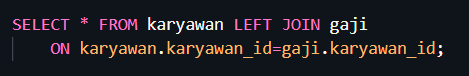
* INNER JOIN

Tipe join ini akan mengambil semua row dari table asal dan table tujuan dengan kondisi nilai key yang terkait saja, dan jika tidak maka row tersebut tidak akan muncul.



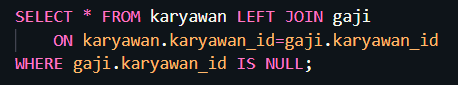
* LEFT JOIN

Biasa juga dikenal dengan LEFT OUTER JOIN merupakan perintah join untuk menampilkan semua data sebelah kiri dari table yang di joinkan dan menampilkan data sebelah kanan yang cocok dengan kondisi join. Jika tidak ditemukan kecocokan, maka akan di set NULL secara otomatis.



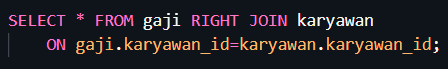
* LEFT JOIN WHERE NULL

LEFT JOIN juga bisa menampilkan data yang hanya kondisi key pada table tamu (foreign key) kosong (NULL).



* RIGHT JOIN

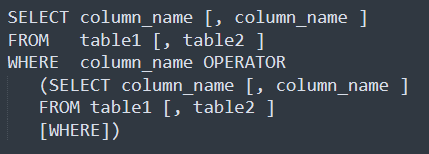
Menampilkan semua data yang ada di table sebelah kanan dan mencari kecocokan key pada table sebelah kiri.



* SUBQUERY

Subquery adalah query dalam query SQL lain dan tertanam dalam klausa WHERE. Sebuah subquery digunakan untuk mengembalikan data yang akan digunakan dalam query utama sebagai syarat untuk lebih membatasi data yang akan diambil.

QUERY



SUBQUERY

**EFFECTIVE QUERY : (Technical & Bussiness Point of view)**

* Gunakan Query yang tidak terlalu panjang.
* Hindari Fungsi yang tidak terlalu penting.
* Tidak selalu bergantung padA *Subquery*.
* Hanya pergunakan kolom yang dibutuhkan / tidak menggunakan (\*) di setiap SELECT.
* Hindari penggunaan Fungsi WHEN jika memungkinkan.
* Selalu pergunakan Fungsi LIMIT jika akan mengecek sebuah table.