**Tugas UAS Basis Data**



Disusun Oleh:

Abdul Rahman Saleh 21254322018

M.Ghozi Syah Putra 21254323029

**Teknologi Rekayasa Komputer 2021**

**POLITEKNIK PERTANIAN NEGERI PAYAKUMBUH**

**I.Proses Normalisasi**

**1. Normal Form (1NF)**

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).

Contoh Normalisasi Database 1NF



dari manual bon pembelian diatas kita dapat menjadi bentuk normal pertama dengan memisah-misahkan data pada atribut-atribut yang tepat dan bernilai atomik, juga seluruh record / baris harus lengkap adanya.

**2. Normal form (2NF)**

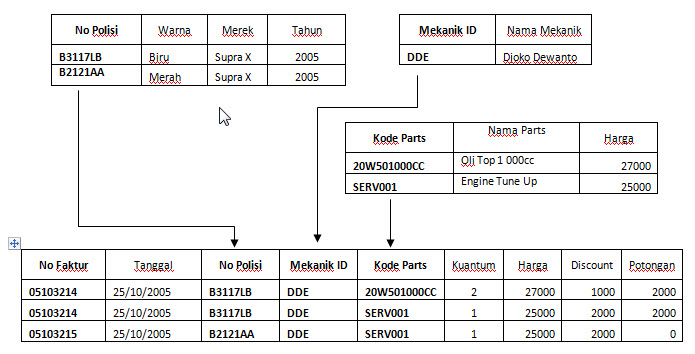
Syarat untuk menerapkan normalisasi bentuk kedua ini adalah data telah dibentuk dalam 1NF, berikut adalah beberapa fungsi normalisasi 2NF.

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.

Contoh normalisasi database bentuk 2NF



Bentuk normal kedua dengan melakukan dekomposisi tabel diatas menjadi beberapa tabel dan mencari kunci primer dari tiap-tiap tabel tersebut dan atribut kunci haruslah unik.

**3. Normal Form (3NF)**

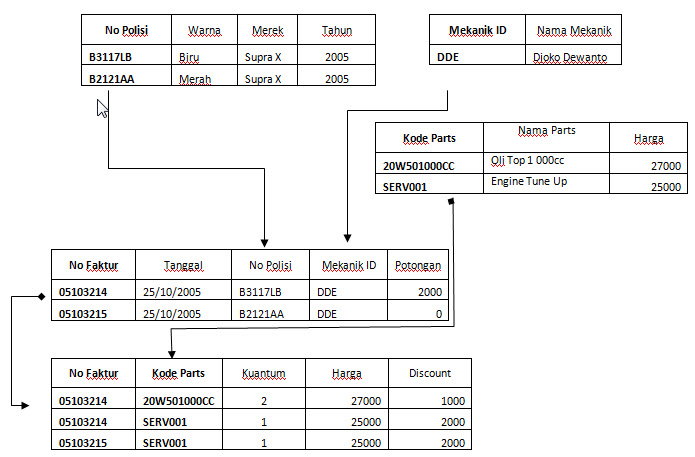
Normalisasi database dalam bentuk 3NF bertujuan untuk menghilangkan seluruh atribut atau field yang tidak berhubungan dengan primary key. Dengan demikian tidak ada ketergantungan transitif pada setiap kandidat key. Syarat dari bentuk normal ketiga atau 3NF adalah :

Memenuhi semua persyaratan dari bentuk normal kedua.

Menghapus kolom yang tidak tergantung pada primary key.

Contoh Normalisasi Database Bentuk 3NF

Bentuk normal ketiga mempunyai syarat, setiap relasi tidak mempunyai atribut yang bergantung transitif, harus bergantung penuh pada kunci utama dan harus memenuhi bentuk normal kedua (2 NF).



BCNF Boyce–Codd normal form

Merupakan sebuah teknik normalisasi database yang sering disebut 3.5NF, memiliki hubungan yang sangat erat dengan bentuk 3NF. Pada dasarnya adalah untuk menghandle anomali dan overlooping yang tidak dapat di handle dalam bentuk 3NF. Normalisasi database bentuk ini tergantung dari kasus yang disediakan, tidak semua tabel wajib di normalisasi dalam bentuk BCNF.

* Suatu relasi disebut memenuhi BCNF jika dan hanya jika setiap determinan yang ada pada relasi tersebut adalah candidate key.
* Definisi yang lain : Suatu relasi disebut memenuhi BCNF jika untuk setiap FD nontrivial : X à A atribut X adalah superkey.
* Untuk normalisasi ke bentuk BCNF, maka tabel 3NF didekomposisi menjadi beberapa tabel yang masing-masing memenuhi BCNF.
* Untuk normalisasi ke bentuk BCNF, maka tabel 3NF didekomposisi menjadi beberapa tabel yang masing-masing memenuhi BCNF.

Contoh:

Diketahui tabel R=(A,B,C) dengan FD : A & B dan B & C maka R bukan BCNF, sebab :

v A Superkey ?

AàB (diketahui)

AàA (refleksif)

Sehingga A & (A,B,C) atau A & R. Jadi A superkey.

v B Superkey ?

B & C (diketahui)

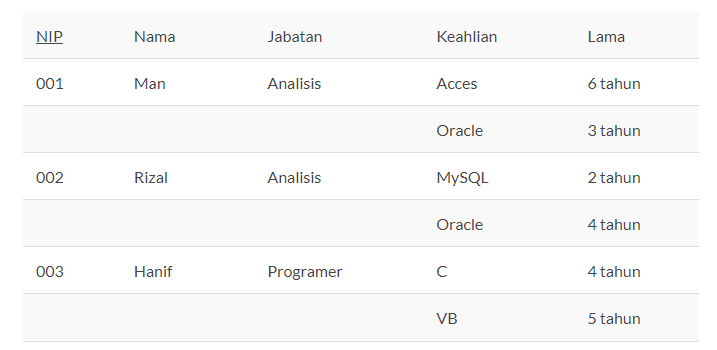
B & B (refleksif)

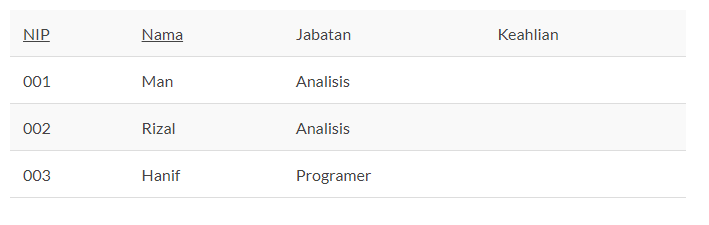
Tapi BA. Sehingga B & A,B,C atau B bukan superkey. Agar R memenuhi BCNF maka didekomposisi menjadi :

R1=(A,B) ; FD : A & B dan

R2=(B,C) ; FD : B & C. Sehingga R1 dan R2 masing-masing memenuhi BCNF. Sebab A dan B dua-duanya sekarang menjadi superkey.

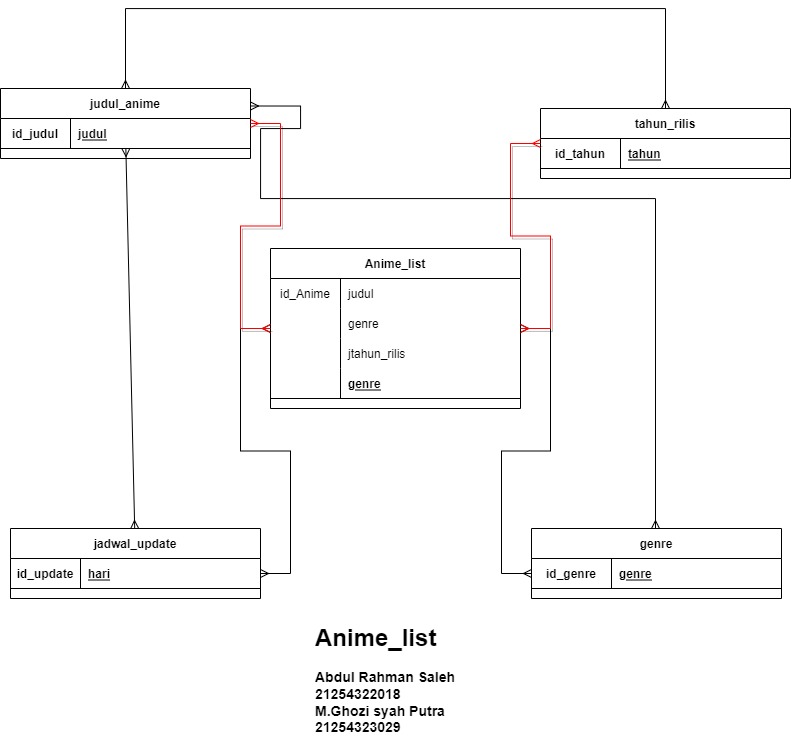
Belum normal







**II.ERD**

****

**III.Proses Permbentukan Struktur Database**

Bisa menggunakan command sebagai berikut untuk membuat database dan tabel di database

**CREATE** **USER** 'uas'@'localhost'**IDENTIFIED** **BY** '123456'

**CREATE** **DATABASE** anime\_list;

**GRANT** **ALL** **PRIVILEGES** **ON** anime\_list.\***TO** 'uas'@'localhost'**IDENTIFIED** **BY** '123456'

**USE** anime\_list;

**CREATE** **TABLE** **IF** **NOT** **EXISTS** jadwal\_update (

id\_update **INT** (50) **NOT** **NULL**,

hari **VARCHAR** (100) **NOT** **NULL**,

**PRIMARY** **KEY** (id\_update)

);

**IV.Contoh query sederhana Insert Select**

**a.Insert**

**INSERT** **INTO** tahun(id\_tahun,tahun) **VALUES** (31,1999);

**INSERT** **INTO** tahun(id\_tahun,tahun) **VALUES** (32,2001);

**INSERT** **INTO** tahun(id\_tahun,tahun) **VALUES** (33,2004);

**INSERT** **INTO** tahun(id\_tahun,tahun) **VALUES** (34,2012);

**INSERT** **INTO** tahun(id\_tahun,tahun) **VALUES** (35,2017);

**b.Select**

**SELECT** b.id\_author, b.title, a.first\_name, a.last\_name

**FROM** books **AS** b

**V.Contoh join dan penjelasan**

Perintah JOIN dalam SQL digunakan untuk menampilkan data pada table yang saling berelasi atau tanpa berelasi tapi berhubungan . Artinya kita dapat menampilkan data dalam beberapa table dengan melihat ada kesamaan antar tabel walau ada entitas yang berbeda namun isinya dapat kita hubungkan.

INNER JOIN yaitu sebuah perintah yang digunakan untuk menampilkan sebuah data atau record dengan menghubungkan dua table atau lebih dalam satu perintah, pada perintah inner join data yang akan ditampilkan hanyalah data yang memiliki pasangan saja, maksudnya yaitu data pada table tidak memiliki sebuah kesamaan maka data tidak akan ditampilkan.

Ketika memasukkan query berikut.

**SELECT** b.id\_author, b.title, a.first\_name, a.last\_name

**FROM** books **AS** b

**INNER** **JOIN** author **AS** a

**ON** b.id\_author = a.id\_author

**ORDER** **BY** b.id\_author;

Hasil yang ditampilkan dari tabel sebagai berikut:

