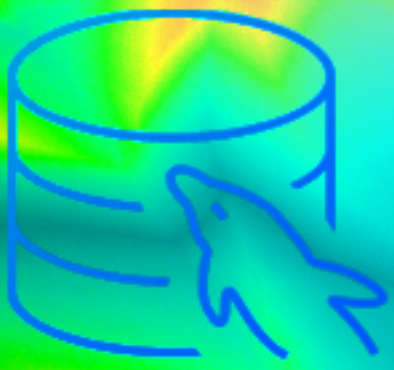
Adityo Permana W

MODUL

Perancangan Basis Data



Nama :

NPM :

Program Studi Sistem Informasi
Program Diploma Tiga
Program Diploma
Universitas Teknologi Yogyakarta

2022

MODUL PRAKTIKUM

PERANCANGAN BASIS DATA

Disusun Oleh:

Adityo Permana Wibowo, S.Kom., M.Cs.

Program Studi Sistem Informasi Program Diploma Tiga
Program Diploma
Universitas Teknologi Yogyakarta
2021

@2021
Diterbirkan Oleh:
Universitas Teknologi Yogyakarta
Jl. Siliwangi, Jombor, Sleman, Yogyakarta
Email : publikasi@uty.ac.id
Website : uty.ac.id

Perancangan Basis Data

ISBN:
00000000000000

Oleh:
Adityo Permana Wibowo

Edisi ke-1
Cetakan Pertama, 2021

PRAKATA

Perkembangan teknologi dirasa sangat cepat sekali, salah satunya adalah perkembangan teknologi basis data. Pembangunan teknologi basis data, dibutuhkan proses perancangan basis data sampai dengan implementasi basis data pada sebuah DBMS. Pembahasan perancangan basis data meliputi penggunaan SQL sampai dengan NoSQL.

Modul Perancangan Basis Data ini dibuat untuk digunakan pada mata kuliah Perancangan Basis Data Praktikum di Program Studi Sistem Informasi Program Diploma Tiga. Fokus pembelajaran modul ini adalah merancang basis data mulai dari penentuan entitas sampai dengan implementasinya di DBMS yang berbasis SQL maupun NoSQL. Modul ini menggunakan studi kasus yang menjadi trend saat ini sehingga mahasiswa bisa lebih mudah memahami prinsip-prinsip dasar dalam merancang dan mengimplementasikan basis data. Modul ini juga menyertakan tugas akhir (final project) untuk mahasiswa agar bisa dijadikan sebagai pembelajaran untuk mengembangkan aplikasi selanjutnya.

Secara garis besar, materi dalam modul terbagi menjadi 4 bagian yaitu pembuatan desain ER-Diagram, konversi ER-Diagram menjadi tabel, implementasi ke DBMS yang berbasis SQL, dan implementasi basis data ke DBMS yang berbasis NoSQL.

Akhir kata, penyusun mengucapkan terimakasih kepada semua pihak yang telah membantu menyelesaikan modul ini. Kritik dan saran sebagai penyempurnaan modul ini sangat kami harapkan bisa dikirimkan melalui email adityopw@uty.ac.id.

Yogyakarta, Desember 2021

Adityo Permana Wibowo, S.Kom., M.Cs.

DAFTAR ISI

PENDAHULUAN

I. DESKRIPSI MATERI

Materi Perancangan Basis Data di modul ini mengambil refrensi dari buku yang ditulis oleh Abraham Sylberschat, dengan judul buku Database System Concept. Buku tersebut diterbitkan tahun 2011.

Materi yang dibahas di dalam modul ini mulai dari penentuan entitas dan atribut, pembuatan aturan bisnis, penggambaran ERD, penentuan key/kunci basis data, implementasi query mulai dari DDL, DML, dan DCL. Modul ini juga membahas implementasi DBMS berbasis NoSQL, yaitu Firebase.

Masing-masing materi disertakan berbagai macam studi kasus sehingga diharapkan materi bisa lebih dimengerti oleh mahasiswa. Pemilihan perangkat lunak DBMS yang berbasis SQL dan NoSQL tersebut dikarenakan banyak digunakan di dunia kerja, sehingga mahasiswa disiapkan untuk pemahaman terkait perangkat lunak DBMS tersebut.

Setelah mempelajari modul ini diharapkan mahasiswa dapat mempunyai landasan atau struktur mengenai perancangan basis data dan implementasinya di DBMS yang berbasis SQL dan NoSQL. Mengkoneksikan basis data dengan program aplikasi baik itu untuk aplikasi berbasis web ataupun aplikasi berbasis mobile sehingga mahasiswa bisa lebih siap untuk memecahkan sebuah kasus atau permasalahan.

II. PRASYARAT

Sebelum menggunakan modul ini diharapkan mahasiswa sudah memenuhi beberapa prasyarat, antara lain:

a) Mahasiswa mampu menggunakan *command prompt* meskipun berbasis windows.
b) Mahasiswa mempunyai pemahaman logika yang baik (sudah menempuh matakuliah Algoritma dan Struktur Data).
c) Mahasiswa sudah lulus matakuliah Sistem Informasi.
d) Mahasiswa sudah lulus matakuliah pemrograman.

III. PETUNJUK PENGGUNAAN MODUL

Modul ini dapat digunakan mahasiswa dengan pertimbangan sebagai berikut:

a) Mahasiswa telah memiliki modul dan telah membaca modul sebelum matakuliah praktikum dimulai.

b) Mahasiswa mempelajari serta mengidentifikasi isi modul yang diuraikan lebih rinci oleh asisten praktikum.
c) Mahasiswa dan asisten praktikum mendiskusikan materi untuk mencari penyelesaian terhadap kasus tertentu.
d) Mahasiswa menyimpulkan isi materi yang didiskusikan.
e) Mahasiswa menjawab soal latihan yang diberikan.
f) Pemberian pengayaan materi bagi mahasiswa yang telah memahami dan menyelesaikan soal latihan.
g) Memberikan tinjauan ulang terhadap materi sekaligus mengidentifikasi kesulitan-kesulitan mahasiswa dalam memahami materi.

IV. STANDAR KOMPETENSI

a) Mahasiswa mampu menghasilkan rancangan basis data yang paling optimal dan tidak ada redudansi.
b) Mahasiswa mampu memecahkan permasalahan basis data menggunakan perintah SQL.
c) Mahasiswa mampu memecahkan permasalahan basis data terkait pengelolaan user dan hak akses masing-masing user.
d) Mahasiswa mampu menerapkan basis data dengan konsep NoSQL.

BAB 1. PERSIAPAN PRAKTIKUM DAN PENGANTAR BASIS DATA

1.1. KOMPETENSI DASAR

Setelah mempelajari Bab ini, mahasiswa:

1) Memahami peraturan kegiatan praktikum di lab computer
2) Memahami teknis persiapan praktikum basis data di lab komputer
3) Memahami konsep dasar dan lingkungan basis data

1.2. INDIKATOR

Setelah mempelajari Bab ini, mahasiswa mampu:

1) Mentaati peraturan kegiatan praktikum di lab komputer
2) Mengikuti aturan teknis persiapan praktikum di lab komputer
3) Menjelaskan konsep dasar dan lingkungan basis data

1.3. URAIAN MATERI

1.2.1. Peraturan Praktikum

1) Praktikum diampu oleh Dosen Mata Kuliah Praktikum dan dibantu oleh Asisten Praktikum.
2) Praktikum dilaksanakan di Laboratorium komputer sesuai jadwal yang sudah ditentukan.
3) Mahasiswa wajib membawa modul praktikum, kartu praktikum, dan alat tulis, serta mengisi daftar hadir
4) Durasi kegiatan dalam 1 sks praktikum sama dengan 100 menit, sehingga jika matakuliah berbobot 4 SKS, maka total pertemuan 400 menit yang dibuat dalam 2 pertemuan dalam satu minggu.
5) mahasiswa wajib hadir minimal 75% dari seluruh pertemuan praktikum di lab. Jika total kehadiran kurang dari 75% maka nilai Mata Praktikum = 0.
6) Praktikan yang datang terlambat lebih dari 15 menit: tidak diperbolehkan mengikuti praktikum
7) Saat praktikum berlangsung, asisten praktikum dan mahasiswa:
   - Wajib mematikan/ men-silent semua alat komunikasi.
   - Dilarang membuka aplikasi yang tidak berhubungan dengan praktikum yang berlangsung.
   - Dilarang mengubah setting software maupun hardware komputer tanpa ijin.
   - Dilarang membawa makanan maupun minuman di ruang lab komputer.

- Dilarang memberikan hasil praktikum ke peserta praktikum lain.
- Dilarang membuang sampah/sesuatu apapun di dalam ruangan lab komputer.

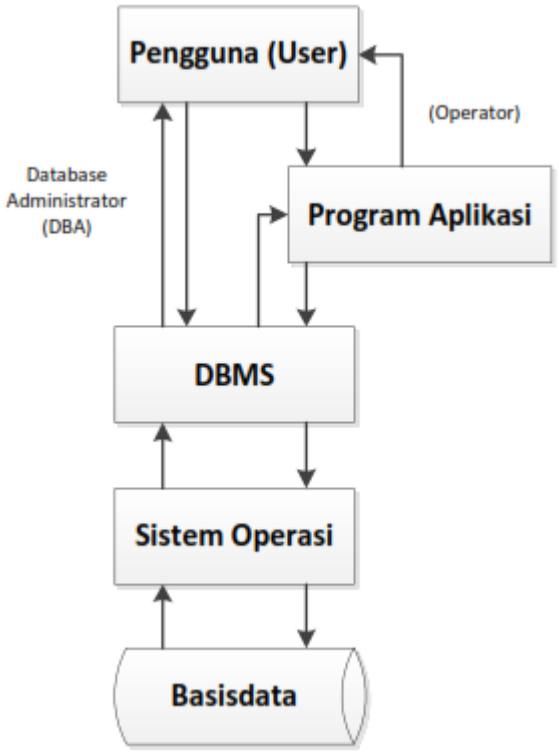
1.2.2. Persiapan sebelum Praktikum

Persiapan yang harus dilakukan sebelum praktikum, antara lain:

1) Pastikan computer yang digunakan berjalan lancer
2) Aplikasi yang digunakan sudah terinstal di komputer yang akan digunakan untuk praktikum
3) Aplikasi yang digunakan antara lain: DBMS MySQL, Microsoft Visio.
4) Materi yang digunakan di Praktikum adalah lanjutan materi dari mata kuliah teori.

1.2.3. Pengantar Basis data

Basis data merupakan sebuah media disebuah perangkat komputer yang digunakan untuk menyimpan data. Dalam proses penyimpanan, basis data melewati beberapa komponen perangkat komputer yang terdiri dari pengguna (DBA dan Operator), Program Aplikasi, DBMS, dan sistem operasi. Pengguna (DBA) bisa mengakses basis data melewati DBMS, sementara pengguna (operator) tidak bisa melewati DBMS langsung, tetapi melewati program aplikasi. Dari program aplikasi dan DBMS akan melewati sistem operasi yang kemudian diteruskan ke basis data. Komponen perangkat komputer tersebut disebut juga dengan lingkungan basis data. Komponen perangkat komputer untuk basis data terlihat pada Gambar 1.1



Gambar 1.1 Struktur Basis Data pada Perangkat Komputer

1.2.4. Kriteria Basis Data

Kriteria yang ada pada basis data:

a) Berorientasi pada data dan bukan pada program yang akan menggunakannya.
b) Dapat digunakan oleh orang yang berbeda-beda atau program aplikasi tanpa perlu mengubah basis data.
c) Data dalam basis data dapat berkembang dengan mudah baik volume maupun strukturnya.
d) Data yang ada dapat memenuhi kebutuhan sistem baru secara mudah.
e) Data dapat digunakan dnegan cara yang berbeda-beda.
f) Minimalisir kerangkapan data.

1.2.5. Aspek Penting Basis Data

Aspek penting dalam basis data yang harus diperhatikan :

a) Kerangkapan data = munculnya data yang secara berulang kali pada file basis data yang semestinya tidak diperlukan.
b) Inkonsistensi data = munculnya data yang tidak konsisten pada kolom yang sama dalam satu atau beberapa file data yang dihubungkan.

c) Data terisolasi = program aplikasi tidak dapat mengakses data-data tertentu yang ada pada basis data.
d) Keamanan data = perlindungan data dalam basis data.
e) Integritas data

1.4. SOAL LATIHAN

Jawab pertanyaan dibawah ini :

1) Jelaskan pengertian sistem basis data menurut anda, (sertakan sumber jika mengutip dari pernyataan orang lain).
2) Menurut anda, seberapa pentingkah penggunaan basis data saat ini? Berikan contoh pentingnya penggunaan basis data.
3) Jelaskan perkembangan basis data menurut anda saat ini.

BAB 2. ER-DIAGRAM

2.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami fungsi dan penggambaran Entitas dalam basis data.
2) Mehamami fungsi dan penyusunan aturan bisnis dalam basis data.
3) Memahami konsep kardinalitas dan hubungan antar entitas dalam basis data.
4) Memahami konsep penggambaran ER-Diagram

2.2. INDIKATOR

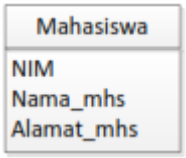
Setelah mempelajari bab ini, mahasiswa mampu:

1) Menentukan entitas dalam basis data.
2) Menentukan aturan bisnis dalam basis data.
3) Menentukan kardinalitas dan hubungan antar entitas dalam basis data
4) Menghasilkan gambar ER-Diagram sesuai kaidah yang ditentukan

2.3. URAIAN MATERI

2.3.1. Entitas

Entitas merupakan pernyataan untuk sebuah objek dalam penggambaran ER-Diagram. Entitas memiliki sifat dan nilai yang unik yang disebut dengan atribut. Sehingga yang dinyatakan sebagai entitas adalah objek yang memiliki sifat dan nilai unik (atribut). Contoh, mahasiswa dalam sebuah universitas adalah sebuah entitas karena memiliki sifat dan nilai unik. Nilai unik yang dimiliki seorang mahasiswa adalah NIM (Nomor Induk Mahasiswa) yang bisa digunakan juga untuk mengidentifikasi seorang mahasiswa. Dalam pembuatan basis data entitas ini nantinya akan menjadi sebuah tabel. Simbol penggambaran entitas terdiri dari 2 bagian. Bagian atas menunjukkan nama entitas, sedangkan bagian bawahnya disebut atribut. Atribut merupakan bagian/rincian dari entitas dimana atribut tersebut akan menjadi kolom pada relasi tabel. Simbol entitas terlihat pada Gambar 2.1.



Gambar 2.1 Simbol Entitas

2.3.2. Relasi Antar Entitas

Hubungan relasi adalah pernyataan yang menghubungkan antara satu entitas dengan entitas yang lainnya. Dalam basis data, hubungan relasi ini nantinya yang menentukan relasi antar tabel. Hubungan relasi digambarkan dengan simbol belah ketupat seperti pada Gambar 2.2.

Gambar 2.2 Simbol Relasi hubungan antar entitas

2.3.3. Aturan Bisnis

Aturan bisnis atau biasa juga disebut dengan Business Rule, biasa digunakan untuk membantu pembuatan ER-Diagram dalam merancang basis data. Penyusunan aturan bisnis nantinya juga bisa digunakan untuk penentuan kardinalitas antar entitas. Ketentuan pembuatan aturan bisnis yaitu:

1) Setiap objek dinyatakan sebagai satu kesatuan yang paling kecil
2) Hubungan antar entitas menggunakan kata kerja aktif

2.3.4. Kardinalitas

Pemetaan kardinalitas menunjukkan jumlah hubungan relasi antara satu entitas dengan entitas lainnya. Pemetaan kardinalitas berfungsi untuk menentukan relasi antar tabel di basis data. Pemetaan kardinalitas terdiri dari:

1) Relasi satu ke satu (*one to one*)

Relasi dari satu ke satu merupakan pernyataan bahwa setiap *record* dalam tabel A hanya dapat memiliki satu *record* yang bersesuaian dalam tabel B, dan sebaliknya. Jenis relasi ini tidak umum karena sebenarnya tabel A dan tabel B dapat digabungkan menjadi satu tabel. Relasi ini dapat digunakan untuk membagi tabel yang memiliki *field* yang banyak, untuk mengisolasi sebagian tabel dengan alasan keamanan data. Simbol relasi *one to one* terlihat pada Gambar 2.3.

Gambar 2.3 Kardinalitas *one to one*

2) Relasi satu ke banyak (*one to many*)

Relasi satu ke banyak adalah jenis relasi yang paling umum. Dalam relasi satu ke banyak menyatakan sebuah record dalam tabel A dapat memiliki banyak record bersesuaian dalam tabel B. tetapi sebuah record dalam tabel B hanya memiliki sebuah record yang bersesuaian dalam table A. Simbol relasi one to many terlihat pada Gambar 2.4.

Gambar 2.4 Kardinalitas *One to Many*

3) Relasi banyak ke satu (many to one)

Sama halnya dengan relasi satu ke banyak, relasi banyak ke satu menyatakan beberapa record dalam tabel A memiliki sebuah record yang bersesuaian dalam table B. Sebaliknya, record dalam tabel B dapat memiliki banyak record bersesuaian dalam tabel A. Simbol relasi many to one terlihat pada Gambar 2.5.

Gambar 2.5 Kardinalitas *Many to One*

4) Relasi banyak ke banyak (*many to many*)

Dalam relasi banyak ke banyak, sebuah record dalam tabel A dapat memiliki banyak record yang bersesuaian dalam tabel B, dan sebuah record dalam tabel B dapat memiliki banyak record yang bersesuaian dala tabel A. Jenis relasi ini hanya dimungkinkan jika kita mendefinisikan tabel baru sebagai perantara. Relasi banyak ke banyak sebenarnya merupakan dua buah relasi satu ke banyak terhadap tabel perantara. Simbol many to many terlihat pada Gambar 2.6.

Gambar 2.6 Kardinalitas Many to Many

2.4. SOAL LATIHAN

Buatlah rancangan ERD dari **Sistem Informasi Pengolahan Nilai Mata kuliah**. Inputan dari aplikasi pengolahan nilai mata kuliah adalah bisa melakukan pendataan mahasiswa, dosen, matakuliah, petugas akademik. Proses yang terjadi di dalam aplikasi transaksi pengolahan nilai. Luaran yang dihasilkan oleh aplikasi adalah daftar nilai mahasiswa berdasarkan mata kuliah. Rancangan yang harus anda buat meliputi:

a) Penentuan entitas yang terlibat.
b) Penentuan aturan bisnis dari masing-masing entitas
c) Hubungan antar entitas beserta kardinalitasnya

BAB 3. KONVERSI ER-DIAGRAM MENJADI TABEL

3.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami konsep penentuan key.
2) Memahami aturan konversi ER-Diagram menjadi Tabel.
3) Memahami aturan penerapan tabel ke dalam DBMS

3.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menentukan tipe key dan atribut/kolom yang bertindak sebagai key.
2) Menghasilkan relasi tabel berdasarkan ER-Diagram sesuai dengan aturan yang berlaku.
3) Menghasilkan database pada sebuah DBMS.

3.3. URAIAN MATERI

3.4. Key

Sebuah entitas mempunyai beberapa himpunan. Himpunan tersebut biasa disebut dengan atribut. Dalam perspektif database, atribut berfungsi sebagai pembeda antara satu entitas dengan entitas lainnya. Oleh karena itu nilai-nilai atribut dari suatu entitas harus mempunyai keunikan tersendiri. Dengan demikian untuk membuat nilai atribut berbeda dibuatlah key. Key ini nanti yang akan menjadikan perbedaan antar entitas. Yang menjadi key adalah satu atau beberapa atribut yang mempunyai nilai paling unik dengan atribut lainnya. Jenis-jenis key terdiri dari:

a) Simple Key (SK):

Kunci relasi yang tersusun atas sebuah atribut, terjadi bila sifat unik telah dapat terpenuhi dengan menggunakan sebuah atribut saja.

b) Composite Key:

Kunci yang tersusun atas gabungan atribut, terjadi bila tidak dipenuhi oleh sebuah atribut tetapi harus menggabungkan lebih dari satu/beberapa atribut.

c) Candidate Key:

Satu atau gabungan atribut yang bersifat unik yang dapat digunakan untuk membedakan setiap record dalam relasi.

d) Primary Key (PK):

Salah satu dari Candidate Key yang dipilih sebagai kunci utama untuk membedakan setiap record dalam relasi. Kolom yang memiliki nilai paling unik bisa dinyatakan sebagai Primary Key (Kunci Utama). Field yang menjadi Primary

Key pada satu tabel bisa menjadi kunci pada tabel lain, yang disebut Foreign Key (Kunci Tamu).

e) **Foreign Key (FK):**
Satu/gabungan sembarang atribut yang menjadi Primary Key pada suatu tabel dan menjadi key di tabel lainnya.

f) **Alternate Key (AK):**
Bagian dari Candidate Key yang tidak dipilih sebagai Primary Key. Dalam relasi tidak harus mempunyai Alternate Key, bergantung pada jumlah Candidate Key yang ada.

3.5. Konversi ER-Diagram menjadi Tabel

Setelah ER-Diagram berhasil dibuat, Langkah selanjutnya yang harus dilakukan dalam perancangan basis data adalah melakukan konversi dari ER-Diagram menjadi tabel. Aturan konversi ER-Diagram menjadi tabel bisa digunakan dengan cara mengacu dari kardinalitas yang sudah ditentukan antar entitasnya. Beberapa aturan konversi antara ER-Diagram menjadi tabel, seperti terlihat pada Tabel 3.1.

Tabel 3.1 Aturan konversi ER-Diagram menjadi tabel

No.	Dari ER-Diagram	Menjadi Tabel
1	kardinalitas one to many atau many to one	Akan ada FK di tabel yang terhubung many
2	Kardinalitas one to one	Bisa terjadi terhubung pada 1 tabel yang sama, atau juga bisa terhubung pada salah satu tabel bentukan.
3	Kardinalitas many to many	Akan ada satu tabel bentukan yang berfungsi untuk merelasikan antara satu tabel dengan tabel yang lainnya. Tabel tersebut tidak bisa langsung terhubung.

3.6. SOAL LATIHAN

Berdasarkan rancangan ER-Diagram Aplikasi Sistem Informasi Pengolahan Nilai Mata Kuliah yang sudah dibuat pada poin 2.4 (Bab 2), konversikan ER-Diagram tersebut menjadi tabel sehingga menghasilkan desain yang siap untuk diimplementasikan ke dalam DBMS.

BAB 4. DDL (DATA DEFINITION LANGUAGE)

4.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian dan fungsi query DDL.
2) Memahami konsep penggunaan constraint.

4.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan dan mengimplementasikan query DDL.
2) Menghasilkan komposisi tabel dengan constraint yang tepat.

4.3. URAIAN MATERI

DDL (Data Definition Language) merupakan perintah dasar dalam SQL yang digunakan untuk membuat basis data dan mendefinisikan struktur tabel dalam basis data tersebut. Struktur tabel yang dimaksud terdiri dari nama kolom, lebar kolom, penentuan key, dan relasi antar tabel. Statement query DDL adalah kumpulan perintah terdiri dari Create, Alter, Drop untuk mendefinisikan tipe data dari objek-objek basis data. Objek basis data (pada MySQL) tersebut terdiri dari database, table, view, index, stored procedure, function, trigger. Perintah-perintah DDL. seperti terlihat pada Tabel 4.1.

Tabel 4.1 Perintah DDL

Create (Pembuatan)	**Alter/Rename (Perubahan)**	**Drop (Penghapusan)**
Create Database	Alter Database	Drop Database
Create Function	Alter Function	Drop Function
Create Index	Alter Procedure	Drop Index
Create Procedure	Alter Table	Drop Procedure
Create Table	Alter View	Drop Table
Create Trigger	Rename Table	Drop Trigger
Create View		Drop View
Create Stored Procedure		Drop Stored Procedure

4.3.1. Perintah Query Create

Penggunaan perintah Create biasanya digunakan untuk pembuatan database dan tabel. Perintahnya seperti terlihat di bawah ini.

a) Membuat Database:

Perintah dasar DDL untuk membuat database seperti di bawah ini:

```
CREATE DATABASE <nama_database>;
```

Penggunaan perintah create tabel seperti terlihat di bawah ini:

```
CREATE DATABASE akademik;
```

b) Membuat Tabel:

Perintah dasar DDL untuk membuat tabel seperti di bawah ini:

```
CREATE TABLE <nama_tabel>, (<field_1> <tipe data>(size), <field_2> <tipe data>(size), <field_3> <tipe data>(size));
```

Penggunaan perintah create tabel seperti terlihat di bawah ini:

```
CREATE TABLE ms_karyawan (
      kd_cbg varchar (10) not null,
      kd_kary varchar (10) not null,
      nm_depan varchar (8) default null,
      nm_blkg varchar (9) default null,
      jns_kelamin varchar (8) default null,
primary key (kd_cbg))
```

4.3.2. Perintah Query Alter/Rename

Penggunaan perintah Alter/Rename biasanya digunakan untuk merubah komposisi tabel. Komposisi yang dimaksud antara lain, penambahan dan pengurangan kolom/field, merubah nama tabel dan nama kolom, merubah tipe data dari suatu kolom. Penggunaan perintah Query Alter/rename seperti terlihat di bawah ini. Perintah menambahkan field seperti terlihati di bawah ini.

```
ALTER TABLE <nama tabel> ADD <nama field><type data>;
```

Perintah melakukan perunahan nama kolom bisa menggunakan perintah di bawah ini.

```
ALTER TABLE <nama tabel> ALTER COLUMN <nama field lama> to <nama field baru>;
```

4.3.3. Perintah Query Drop

Penggunaan perintah Drop biasanya digunakan untuk menghapus bagian dari komposisi tabel. Drop biasanya digunakan untuk menghapus tabel dan atau database. Perintah untuk menghapus tabel seperti di bawah ini.

```
DROP TABLE <nama_tabel>;
```

Sedangkan perintah untuk menghapus database, seperti terlihat di bawah ini.

```
DROP DATABASE <nama_database>;
```

4.3.4. Constraint

Constraint adalah aturan atau batasan tertentu pada field/kolom dari sebuah tabel yang diperlukan untuk membuat sebuah relationship. Pengaturan Constraint dibuat untuk mencegah penghapusan data dari suatu tabel yang mempunyai keterkaitan dengan tabel yang lain. Field/kolom yang dimaksud memiliki batasan integritas. Batasan integritas adalah suatu kondisi yang harus bernilai benaruntuk suatu instance dalam basis data, diantaranya adalah: NOT NULL, PRIMARY KEY, FOREIGN KEY, UNIQUE.

Contoh penggunaan Constraint:

a) **Constraint NOT NULL**

Suatu kolom yang didefinisikan NOT NULL berarti kolom tersebut harus terisi, tidak boleh kosong. Jika kolom tersebut didefinisikan sebagai PRIMARY KEY maka secara otomatis NOT NULL.

```
CREATE TABLE employees (
employee_id NUMBER(6),
name varchar(30) NOT NULL,
address varchar(40));
```

b) Constraint PRIMARY KEY

Suatu kolom didefinisikan sebagai PRIMARY KEY, maka membentuk key yang unik pada suatu tabel. Kolom tersebut akan mengidentifikasi suatu baris data menjadi unik.

```
CREATE TABLE employees (
employee_id NUMBER(6) PRIMARY KEY,
name varchar(30) NOT NULL,
address varchar(40));
```

Atau

```
CREATE TABLE employees (
employee_id NUMBER(6),
name varchar(30),
address varchar(40),
CONSTRAINT emp_id PRIMARY KEY (employee_id));
```

c) Constraint FOREIGN KEY

Constraint FOREIGN KEY didefinisikan pada suatu kolom suatu tabel dimana kolom tersebut menjadi PRIMARY KEY pada tabel lain.

```
CREATE TABLE employees (
employee_id NUMBER(6),
name varchar(30) NOT NULL,
address varchar(40),
...
departement_id
CONSTRAINT emp_dept_id FOREIGN KEY (departement_id)
References departement(departement_id));
```

d) Constraint UNIQUE

Constraint UNIQUE digunakan untuk kolom bersifat unik.

```
CREATE TABLE employees (
employee_id NUMBER(6),
name varchar(30) NOT NULL,
address varchar(40),
email varchar(20),
...
CONSTRAINT emp_email UNIQUE (email));
```

4.4. SOAL LATIHAN

Jawab soal Latihan berikut menggunakan sintak query, buatlah sebuah database kemudian buat minimal 3 tabel di dalam database tersebut yang saling terhubung (relasi tabel). Tentukan tipe data field dan field size nya sesuai dengan kebutuhan yang dibuat. Tentukan pula constraint dari masing-masing tabel

BAB 5. DML (DATA MANIPULATION LANGUAGE)

5.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian dan fungsi query DML.
2) Memahami pengertian dan fungsi query Select.

5.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan dan mengimplementasikan Query DML.
2) Menjelaskan dan mengimplementasikan Query Select.

5.3. URAIAN MATERI

DML (Data Manipulation Language) merupakan perintah dalam SQL yang memungkinkan pengguna untuk memanipulasi data dalam tabel. Yang dimaksud memanipulasi data adalah:

a) Menambah data baru pada suatu tabel.
b) Merubah data yang sudah ada pada suatu tabel.
c) Menghapus data pada suatu tabel.
d) Menampilkan informasi yang disimpan dalam tabel pada sebuah basis data.

DML juga bertujuan untuk memudahkan pemakai untuk mengakses data sebagaimana direpresentasikan oleh model data. Perintah DML pada MySQL terdiri dari Call, Delete, Do, Handler, Insert, Load Data Infile, Replace, Select, Truncate, Update. Hanya saja pada umumnya perintah DML yang biasa digunakan adalah Insert, Update, Delete, Select. Jenis DML terdiri dari:

a) **Prosedural**: mensyaratkan pemakai untuk menentukan data apa yang diinginkan serta bagaimana cara mendapatkannya.
b) **NonProsedural**: membuat pemakai dapat menentukan data apa yang diinginkan tanpa menyebutkan bagaimana cara mendapatkannya.

5.3.1. Perintah Query Insert

Perintah INSERT digunakan untuk memasukkan/menyimpan data baru pada suatu tabel. Perintah dasar Query Insert seperti di bawah ini:

```
INSERT INTO <nama tabel> VALUES ('<isi field_1>',' <isi field_2>',' <isi field_3>',' <isi field_4>',' <isi field_5>');
```

5.3.2. Perintah Query Update

Perintah UPDATE digunakan untuk merubah data baru pada suatu tabel. Perintah dasar Query Update seperti di bawah ini:

```
UPDATE <nama tabel> SET <nama field_1=isi baru_1, nama field_2=isi baru_2,……,
nama field_n=isi baru_n WHERE kriteria> ;
```

Perintah UPDATE, harus disertakan kriteria baris/data/row yang akan diubah karena jika tidak diberikan kriteria maka bisa merubah semua data yang ada pada tabel. Kriteria yang dimaksud adalah mengacu pada PRIMARY KEY di tabel yang akan diubah.

5.3.3. Perintah Query Delete

Perintah DELETE digunakan untuk menghapus data baru pada suatu tabel. Perintah dasar Query Delete seperti di bawah ini:

```
DELETE FROM <nama tabel> WHERE <kriteria>;
```

Perintah DELETE, harus disertakan kriteria baris/data/row yang akan dihapus karena jika tidak diberikan kriteria maka bisa menghapus semua data yang ada pada tabel. Kriteria yang dimaksud adalah mengacu pada PRIMARY KEY di tabel yang akan dihapus.

5.3.4. Perintah Select

Perintah dasar query SELECT hanya digunakan untuk menampilkan data yang ada di dalam suatu tabel. Contoh penggunaannya adalah:

```
SELECT *
FROM nama_tabel;
```

Tanda bintang mengandung arti all/semua.

Jika ingin menampilkan kolom tertentu pada suatu tabel, bisa menggunakan perintah.

```
SELECT nama_kolom1, nama_kolom2, nama_kolom3
FROM nama_tabel;
```

Jika ingin menampilkan kolom tertentu pada suatu tabel dan dengan kriteria tertentu, bisa menggunakan perintah:

```
SELECT nama_kolom1, nama_kolom2, nama_kolom3
FROM nama_tabel
WHERE predikat;
```

Dimana predikat adalah suatu kondisi yang ditentukan oleh user. Predikat bisa melibatkan operasi logika AND, OR, NOT, atau operasi perbandingan =, <, >, <>.

5.4. SOAL LATIHAN

Berdasarkan tabel yang sudah dibuat pada poin 4.4 (bab 4), selanjutnya,

1) inputkan data ke dalam 3 tabel tersebut minimal 5 record. (dengan menggunakan sintak query).
2) Ubah beberapa record data dengan menggunakan sintak query UPDATE.
3) Tampilkan beberapa record dengan menggunakan perintah query SELECT

BAB 6. QUERY FUNGSI

6.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami penggunaan perintah query operator untuk menampilkan data dalam 1 tabel.
2) Memahami penggunaan perintah query Fungsi untuk menampilkan data dalam 1 tabel.

6.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menghasilkan data hasil eksekusi perintah query operator.
2) Menghasilkan data hasil eksekuisi perintah query Fungsi.

6.3. URAIAN MATERI

Terdapat beberapa perintah operator atau fungsi yang biasa digunakan di SQL, antara lain SUM, MIN, MAX, COUNT, AVG, dan Group By. Sebelum membahas tentang operator tersebut, ada baiknya perlu diingat-ingat kembali konsep penggunaan operator dan fungsi yang sudah pernah dibahas pada modul semester sebelumnya. Mulai dari Syntak Dasar SQL tentang SUM. SUM adalah suatu fungsi pada SQL yang digunakan untuk menjumlahkan nilai dari sekumpulan record. Selanjutnya fungsi MIN, MIN adalah suatu fungsi pada SQL yang digunakan untuk mendapatkan nilai terkecil dari sekumpulan record. Kemudian fungsi MAX, MAX adalah kebalikan dari MIN, yaitu fungsi yang digunakan untuk mendapatkan nilai tertinggi dari sekumpulan record. Kemudian penggunaan COUNT. Count adalah suatu fungsi pada SQL yang digunakan untuk mendapatkan jumlah baris atau record dari suatu tabel. Kemudian penggunaan AVG (AVERAGE). AVG adalah suatu fungsi pada SQL yang digunakan untuk mendapatkan nilai rata-rata dari sekumpulan nilai record suatu tabel. Terakhir penggunaan Group BY. Fungsi Group By digunakan untuk menampilkan berdasarkan pengelompokkan sebuah kolom.

6.3.1. Query Aggregate

Fungsi Agregate digunakan untuk melakukan proses perhitungan secara cepat. Yang termasuk dalam fungsi aggregate antara lain SUM, MIN, MAX, AVG (Average), dan COUNT.

Fungsi-fungsi yang bisa digunakan dalam SQL antara lain:

a) MAX → untuk menampilkan jumlah terbesar atau paling banyak

```
SELECT MAX(umur) FROM ms_karyawan;
```

b) MIN → untuk menampilkan jumlah terkecil atau paling sedikit

```
SELECT MIN(umur) FROM ms_karyawan;
```

c) Average/AVG → untuk menampilkan jumlah rata-rata

```
SELECT AVG(umur) FROM ms_karyawan;
```

d) SUM → untuk menampilkan nilai jumlah

```
SELECT SUM(umur) FROM ms_karyawan;
```

e) COUNT → untuk menampilkan nilai jumlah cacah

```
SELECT COUNT(kd_karyawan) FROM ms_karyawan;
```

f) LIKE → untuk menampilkan karakter yang memenuhi syarat Like

```
SELECT nama_kary FROM ms_karyawan WHERE nama_kary
                    LIKE %Ahmad%;
```

g) GROUP BY → untuk menampilkan kelompok record

```
SELECT nama_kary, alamat_kary FROM ms_karyawan
                  GROUP BY alamat_kary;
```

h) DISTINCT → untuk meniadakan duplikasi record

6.3.2. Query Aritmatika

Fungsi aritmatika digunakan untuk menghasilkan perhitungan yang diambil dari record atau perhitungan tersendiri. Yang termasuk dalam fungsi aritmatika antara lain penjumlahan (+), pengurangan (-), perkalian (*), pembagian (/), pembagian sisa (%). Contoh penggunaan querynya seperti di bawah ini.

```
SELECT 18+2 AS JUMLAH, 18-2 AS SELISIH, 18*2 AS
PERKALIAN, 18/2 AS PEMBAGIAN, 18%2 AS MODULUS;
```

6.4. SOAL LATIHAN

Perhatikan tabel di bawah ini.

Tabel Buku

KD_BUKU	JUDUL_BUKU	KD_KARANG	KD_TERBIT	JUMLAH
21	Kalkulus	10	1	10
22	Metode Numerik	11	2	20
23	Sistem Basis Data	12	3	40
24	Pengantar Teknologi Informasi	12	3	41
			row(s) 1 - 4 of 4	

Berdasarkan tabel di atas, jawab pertanyaan berikut menggunakan perintah query:

1) Dengan menggunakan perintah query, inputkan 1 data pada tabel buku. Data tersebut yaitu: Kode Buku = 27; Judul Buku = Informatika Sosial; Kode Pengarang = 12; Kode Penerbit = 3; Jumlah stok buku = 35;
2) Tampilkan judul buku yang jumlah bukunya lebih dari 30 buah.
3) Tampilkan jumlah buku yang diterbitkan oleh penerbit yang mempunyai kode 3.
4) Tampilkan judul buku yang jumlah bukunya terbanyak.

BAB 7. QUERY RELASI MULTI TABEL (Where-And dan Join)

7.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami penggunaan perintah query multi tabel menggunakan konsep WHERE-AND.
2) Memahami penggunaan perintah query multi tabel menggunakan konsep JOIN.

7.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menghasilkan tampilan tabel hasil perintah query WHERE-AND
2) Menghasilkan tampilan tabel hasil perintah query JOIN

7.3. URAIAN MATERI

7.3.1. Query Relasi menggunakan Where-And

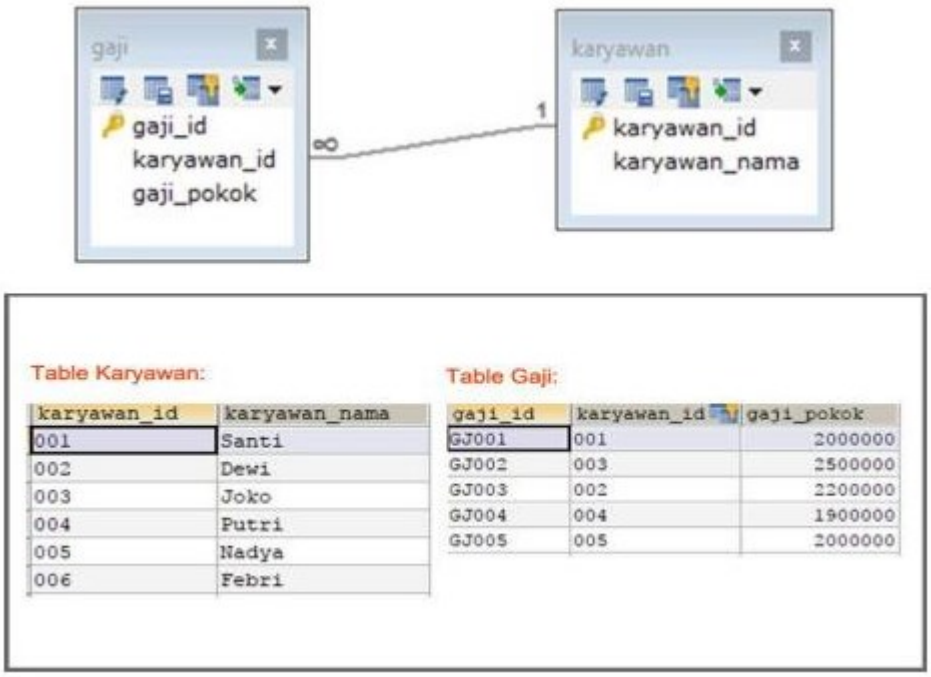
Didalam basis data relasional dimungkinkan untuk mengakses satu atau lebih tabel dalam suatu database dalam waktu bersamaan. Perintah query relasi bisa dilakukan dengan 2 macam perintah, yaitu WHERE-AND dan JOIN. Perintah JOIN sendiri mempunyai beberapa fungsi lagi antara lain INNER JOIN, OUTER JOIN, LEFT JOIN, RIGHT JOIN.

Contoh sintak dasar perintah query Where-And, seperti di bawah ini:

```
SELECT nama_tabel1.nama_field1, nama_tabel2.nama_field2
FROM nama_tabel1, nama_tabel2, nama_tabel3
WHERE nama_tabel1.key1=nama_tabel2.key2 AND
nama_tabel3.key3=nama_tabel2.key2;
```

7.3.2. Query Relasi menggunakan JOIN

Perintah query JOIN umumnya digunakan untuk menampilkan data dari beberapa tabel (relasi tabel). terdapat berbagai macam tipe JOIN antara lain Inner Join, Left Join, Right Join. Sebagai bahan pembelajaran, perhatikan relasi tabel yang terlihat pada Gambar 7.1.

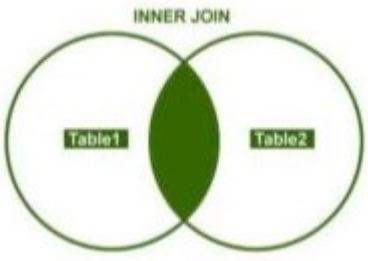


Gambar 7.1 Relasi Tabel Karyawan dan Tabel Gaji

7.3.3. Inner Join

Perintah INNER JOIN merupakan salah satu jenis JOIN yang paling umum digunakan untuk menampilkan *record* dari beberapa tabel. WHERE AND dan INNER JOIN menghasilkan record relasi yang sama. Perbedaan perintah tersebut terletak pada relasinya yaitu ON (untuk JOIN) dan AND (untuk WHERE-AND). 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. Kalau tidak terdapat kondisi key terkait antar table, maka semua row dari kedua table dikombinasikan.

Konsep perintah Inner Join seperti konsep irisan dari dua buah tabel. Jika dibuat diagram venn-nya akan terlihat seperti gambar berikut:



Berdasarkan diagram venn di atas, jika diimplementasikan menggunakan perintah query Inner Join untuk tabel yang terlihat pada Gambar 7.1, seperti di bawah ini:

```
SELECT * FROM karyawan INNER JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id;
```

Perintah query Inner Join tersebut menghasilkan tampilan seperti di bawah ini:

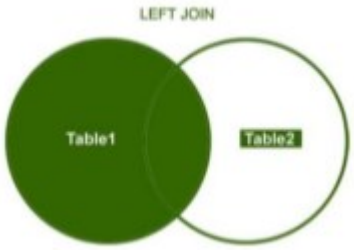
gaji_id	karyawan_id	gaji_pokok	karyawan_id	karyawan_nama
GJ001	001	2000000	001	Santi
GJ002	003	2500000	003	Joko
GJ003	002	2200000	002	Dewi
GJ004	004	1900000	004	Putri
GJ005	005	2000000	005	Nadya

Baris data yang ditampilkan sebanyak 5 baris data, karena INNER JOIN hanya memperhitungkan kondisi key yang terkait antara tabel karyawan dengan tabel gaji. Sedangkan karyawan dengan karyawan_id='006' tidak ditampilkan karena tidak terkait dengan tabel gaji.

7.3.4. Left Join

LEFT JOIN atau 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.

Konsep perintah Left Join seperti konsep irisan dari dua buah tabel. Jika dibuat diagram venn-nya akan terlihat seperti gambar di bawah ini.



Berdasarkan diagram venn di atas, jika diimplementasikan menggunakan perintah query Left Join untuk tabel yang terlihat pada Gambar 7.1, seperti di bawah ini:

```
SELECT * FROM karyawan LEFT JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id;
```

Perintah query Left Join tersebut menghasilkan tampilan seperti di bawah ini:

karyawan_id	karyawan_nama	gaji_id	karyawan_id	gaji_pokok
001	Santi	GJ001	001	2000000
003	Joko	GJ002	003	2500000
002	Dewi	GJ003	002	2200000
004	Putri	GJ004	004	1900000
005	Nadya	GJ005	005	2000000
006	Febri	*(NULL)*	*(NULL)*	*(NULL)*

Selain kondisi diatas, Left Join juga bisa menampilkan data yang hanya kondisi key pada table tamu (*foreign key*) kosong (*NULL*), atau biasa disebut dengan **Left Join Where Null**. Penggunaannya seperti terlihat pada potongan perintah query di bawah ini.

```
SELECT * FROM karyawan LEFT JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id
WHERE gaji.karyawan IS NULL;
```

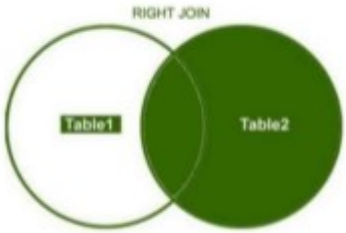
Perintah query di atas menghasilkan tampilan seperti di bawah ini.

karyawan_id	karyawan_nama	gaji_id	karyawan_id	gaji_pokok
006	Febri	*(NULL)*	*(NULL)*	*(NULL)*

7.3.5. Right Join

Kebalikan dari LEFT JOIN adalah RIGHT JOIN, atau biasa juga dikenal dengan RIGHT OUTER JOIN. RIGHT JOIN akan menampilkan semua data yang ada di table sebelah kanan dan mencari kecocokan key pada table sebelah kiri. Jika tidak ditemukan kecocokan, maka akan di set NULL secara otomatis pada table sebelah kiri.

Konsep perintah Right Join seperti konsep irisan dari dua buah tabel, akan tetapi lebih mengacu pada isi data yang ada di tabel sebelah kanan. Jika dibuat diagram venn-nya akan terlihat seperti gambar di bawah ini.



Berdasarkan diagram venn di atas, jika diimplementasikan menggunakan perintah query Right Join untuk tabel yang terlihat pada Gambar 7.1, seperti di bawah ini.

```
SELECT * FROM karyawan RIGHT JOIN gaji ON
karyawan.karyawan_id=gaji.karyawan_id;
```

Perintah query Right Join tersebut menghasilkan tampilan seperti di bawah ini.

gaji_id	karyawan_id	gaji_pokok	karyawan_id	karyawan_nama
GJ001	001	2000000	001	Santi
GJ002	003	2500000	003	Joko
GJ003	002	2200000	002	Dewi
GJ004	004	1900000	004	Putri
GJ005	005	2000000	005	Nadya
(NULL)	*(NULL)*	*(NULL)*	006	Febri

Sama halnya dengan **Left Join Where Null**, Right Join juga bisa menampilkan data yang hanya kondisi key pada table tamu (*foreign key*) kosong (*NULL*), atau biasa disebut dengan **Right Join Where Null**. Penggunaannya seperti terlihat pada potongan perintah query di bawah ini.

```
SELECT * FROM karyawan RIGHT JOIN gaji
ON karyawan.karyawan_id=gaji.karyawan_id
    WHERE gaji.karyawan IS NULL;
```

Perintah query Right Join Where Null tersebut menghasilkan tampilan seperti di bawah ini.

gaji_id	karyawan_id	gaji_pokok	karyawan_id	karyawan_nama
(NULL)	*(NULL)*	*(NULL)*	006	Febri

7.4. SOAL LATIHAN

Perhatikan tabel di bawah ini:

Tabel Buku

KD_BUKU	JUDUL_BUKU	KD_KARANG	KD_TERBIT	JUMLAH
21	Kalkulus	10	1	10
22	Metode Numerik	11	2	20
23	Sistem Basis Data	12	3	40
24	Pengantar Teknologi Informasi	12	3	41
			row(s) 1 - 4 of 4	

Tabel Pengarang

KD_KARANG	PENGARANG	ALAMAT
12	Adityo PW	Sleman
10	Suparman	Sleman
11	Irwanto	JOgja
row(s) 1 - 3 of 3		

Tabel Penerbit

KD_TERBIT	PENERBIT	ALAMAT
1	GaYa Media	Jl. Aspal
2	Ditanam Press	Jl. Aspal Apik
3	Ty0 Publisher	Jl. Brongkalan Boto
row(s) 1 - 3 of 3		

Berdasarkan 3 tabel yang terbentuk (Tabel Buku, Tabel Pengarang, Tabel Penerbit), jawab pertanyaan berikut menggunakan perintah query.

1) Tampilkan judul buku, nama pengarang dan nama penerbit dari masing-masing buku menggunakan query WHERE-AND, INNER JOIN, LEFT JOIN, dan RIGHT JOIN. Kemudian perhatikan perbedaan dari masing-masing hasilnya selanjutnya jelaskan perbedaan hasil tersebut.
2) Tampilkan nama pengarang dan judul buku yang jumlah bukunya lebih dari 30 buah.
3) Tampilkan judul buku yang ditulis oleh pengarang bernama Suparman
4) Tampilkan jumlah buku yang di terbitkan oleh penerbit GaYa Media.

BAB 8. QUERY VIEW

8.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian dan fungsi View di dalam DBMS
2) Memahami konsep pembuatan View di dalam DBMS

8.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan fungsi View di dalam DBMS
2) Menghasilkan View menggunakan perintah Query

8.3. URAIAN MATERI

8.3.1. View

View merupakan salah satu objek basis data yang merepresentasikan sub himpunan dari data yang berasal dari satu atau lebih tabel. View adalah perintah query yang disimpan pada basis data dengan suatu nama tertentu sehingga bisa digunakan setiap saat untuk melihat data tanpa menuliskan ulang query tersebut. View bisa disebut juga dengan tabel virtual.

View digunakan untuk mempermudah penulisan query, menyembunyikan beberapa kolom yang bersifat rahasia serta untuk mempercepat menampilkan data. Pembahasan ini, perintah view terdiri dari create, replace, drop, dan using. Perintah View digunakan untuk menampilkan data dari beberapa tabel, lain halnya jika ingin menyimpan data ke beberapa tabel, yaitu menggunakan Stored Procedure. Stored Procedure merupakan perintah query yang digunakan untuk menyimpan, merubah, dan menghapus data yang disimpan di dalam basis data dengan suatu nama tertentu sehingga bisa dipanggil setiap saat.

8.3.2. Create View

Pembuatan view diawali dengan perintah “Create View” kemudian dilanjutkan nama view, kemudian nama kolom, nama tabel dan disertakan kondisi jika diperlukan. Perintah pembuatan view seperti di bawah ini:

```
CREATE VIEW nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_tabel
WHERE kondisi;
```

8.3.3. Replace View

View yang sudah dibuat, bisa dilakukan perubahan berdasarkan kebutuhan. Misalkan menambahkan kolom, menambah tabel, dan menambahkan filter pada where clause. Untuk merubah view menggunakan sintak seperti di bawah ini:

```
CREATE OR REPLACE VIEW nama_view AS
SELECT kolom_1, kolom_2, kolom_n
FROM nama_tabel
WHERE kondisi;
```

8.3.4. Drop View

Selain bisa dirubah, view yang sudah dibuat bisa dihapus jika sudah tidak dibutuhkan. Untuk menghapus view menggunakan sintak seperti dibawah ini:

```
DROP VIEW nama_view;
```

8.4. STUDI KASUS PEMBUATAN VIEW

8.4.1. Create View

Pada dasarnya view merupakan tabel virtual yang terbentuk dari satu atau beberapa tabel yang ada dalam basis data. Pada studi kasus ini akan dijelaskan pembuatan view untuk 1 tabel dan 2 tabel. Contoh penggunaan sintak create view untuk 1 tabel yaitu:

```
CREATE VIEW v_buku AS
SELECT kd_buku, judul_buku, jml_buku
FROM buku
WHERE jml<=3;
```

Sintak create view di atas menunjukkan pembuatan tabel virtual dari tabel buku yang berisi data buku yang jumlahnya kurang dari 3 buah. Dengan demikian jika ingin mengetahui jumlah buku yang kurang dari 3 cukup memanggil “v_buku” saja, tidak perlu membuat perintah query select lagi.

8.4.2. Replace View

Berdasarkan view yang sudah dibuat di atas (v_kategoribuku), misalkan akan ditambahkan kondisi yaitu jumlah buku yang kurang dari 3 buah. Sehingga perintah viewnya adalah:

```
CREATE OR REPLACE VIEW v_kategoribuku AS
SELECT kd_buku, judul_buku, namakategori
FROM buku, kategori
WHERE  kategori.kd_kategori=buku.kd_kategori,
buku.jml_buku<=3;
```

8.5. SOAL LATIHAN

Buatlah basis data penjualan buku yang terdiri dari tabel:

Buku(kd_buku, judul_buku, kd_pengarang, kd_penerbit, stok)

Pengarang(kd_pengarang, nama_pengarang, alamat_pengarang)

Penerbit(kd_penerbit, nama_penerbit, alamat_penerbit)

Petugas(kd_petugas, nama_petugas, alamat_petugas)

Pelanggan(kd_pelanggan, nama_pelanggan, alamat_pelanggan)

Penjualan(kd_jual, kd_petugas, kd_pelanggan, tanggaljual, total_jual, total_item)

DetailPenjualan(kd_detail, kd_jual, kd_buku, jml_beli)

Berdasarkan tabel yang sudah dibuat, selanjutnya buatlah view untuk menampilkan data dengan ketentuan:

1) Data stok buku berdasarkan penerbit
2) Data jumlah buku berdasarkan pengarang
3) Data penjualan buku yang dilayani oleh petugas tertentu
4) Data buku yang dibeli oleh oleh pelanggan tertentu

BAB 9. STORED PROCEDURE

9.1. KOMPETENSI DASAR

Setelah mempelajari bab ini mahasiswa:

1) Memahami pengertian Stored Procedure pada sebuah DMBS.
2) Memahami fungsi dan penggunaan Stored Procedure pada sebuah DBMS.

9.2. INDIKATOR

Setelah mempelajari bab ini mahasiswa:

1) Mampu menjelaskan pengertian Stored Procedure pada sebuah DMBS.
2) Mampu mengimplementasikan fungsi dan penggunaan Stored Procedure pada sebuah DBMS.

9.3. URAIAN MATERI

Stored Procedure merupakan serangkaian sub routine (baris program) yang dibuat dan disimpan dalam basis data. Stored Procedure memberikan banyak keuntungan bagi developer terutama dari sisi sekuritas basis data dan pengembangan software yang multiplatform serta membutuhkan banyak teamwork.

Stored Procedure berisi kumpulan sintak-sintak SQL yang menghasilkan output tertentu. Stored Procedure bisa mengurangi traffic network dan overhead. Pengelolaan data tidak dilakukan disisi aplikasi, melainkan disisi database server. Penggunaan Stored Procedure antara lain untuk proses manipulasi data (insert, update, dan delete), selain itu juga bisa membatasi akses langsung ke tabel dari basis data.

Perintah stored procedure pada umumnya digunakan untuk proses insert, update dan delete pada sebuah DBMS. Stored Procedure memiliki variable parameter yang dibedakan menjadi:

1) **IN**: Variabel parameter hanya dapat digunakan untuk menerima input saja. IN menjadi nilai default dari variabel parameter
2) **OUT**: Variabel parameter hanya dapat digunakan untuk menyimpan hasil output saja.
3) **INOUT**: Variabel parameter yang digunakan untuk menerima input dan menyimpan hasil output.

9.4. Perintah Dasar Stored Procedure

Setiap perintah stored procedure selalu diawali dan diakhiri dengan fungsi delimiter. Delimiter adalah sebuah tanda batas akhir dari suatu perintah bagi SQL untuk mengeksekusi sebuah perintah query. Secara default delimiter pada sebuah perintah SQL adalah tanda titik koma (;), namun ada juga yang menggunakan garis pipa (|) sebagai batas perintah. Penggunaannya sangat sederhana, sebelum mendefinisikan objek tersebut kita gunakan statement "DELIMITER" diikuti tanda pemisah baru. Setelah di akhir pendefinisian kita kembalikan delimiter lagi kepada tanda titik koma. Sintak dasar perintah stored procedure yaitu:

```
DELIMITER |
CREATE PROCEDURE storedprocedure_name ([proc_parameter[....]])
[characteristic ……] routine_body
```

Keterangan:

- **Proc_parameter**: parameter stored procedure yang terdiri dari IN, OUT, INOUT.
- **Routine_body**: kumpulan perintah stored procedure yang terdiri dari peritnah SQL.

9.5. Studi kasus Pembuatan Stored Procedure

9.5.1. Perintah Stored Procedure untuk Insert

Sebagaimana yang sudah dijelaskan diatas, bahwa stored procedure merupakan kumpulan perintah SQL yang dibuat untuk mempermudah proses yang digunakan di beberapa tempat, sehingga dalam penggunaannya hanya memanggil nama stored procedure-nya saja. Sintak dasar stored procedure insert, yaitu:

```
DELIMITER |
CREATE PROCEDURE tambahbarang
 (IN kd_brg char(10), IN nm_brg varchar(40), IN hrg_sat int(5),
IN sat varchar(5), IN stok int(3))

Begin
 INSERT INTO barang (kode_barang, nama_barang, harga_satuan,
 satuan, jml_barang) values (kd_brg, nm_brg, hrg_sat, sat,
 stok)
End |
DELIMITER;
```

9.5.2. Perintah Stored Procedure untuk Update

Sama halnya stored procedure insert, perintah stored procedure untuk update yaitu:

```
DELIMITER |
CREATE PROCEDURE ubahbarang
 (IN kd_brg char(10), IN nm_brg varchar(40), IN hrg_sat int(5),
IN sat varchar(5), IN stok int(3))

Begin
 UPDATE barang SET nm_brg=@nama_barang, hrg_sat=@harga_satuan,
 sat=@satuan, stok=@jml_barang WHERE kd_brg=@kode_barang;
End |
DELIMITER;
```

9.5.3. Perintah Stored Procedure untuk Delete

Sama halnya stored procedure update, perintah stored procedure untuk delete yaitu:

```
DELIMITER |
CREATE PROCEDURE hapusbarang
 (IN kd_brg char(10), IN nm_brg varchar(40), IN hrg_sat int(5),
IN sat varchar(5), IN stok int(3))

Begin
 Delete FROM barang WHERE kd_brg=@kode_barang;
End |
DELIMITER;
```

9.6. SOAL LATIHAN

Perhatikan tabel berikut:

KD_BUKU	JUDUL_BUKU	KD_KARANG	KD_TERBIT	JUMLAH
21	Kalkulus	10	1	10
22	Metode Numerik	11	2	20
23	Sistem Basis Data	12	3	40
24	Pengantar Teknologi Informasi	12	3	41
			row(s) 1 - 4 of 4	

Buatlah perintah Stored Procedure untuk menambah data, merubah data, dan menghapus data

BAB 10. TRIGGER

10.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian dan fungsi trigger pada MySQL
2) Memahami penggunaan trigger pada MySQL

10.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan pengertian dan fungsi trigger pada MySQL
2) Mengimplementasikan trigger pada MySQL

10.3. URAIAN MATERI

10.3.1. Pengertian Trigger

Trigger adalah object basis data yang berhubungan langsung dengan tabel dan akan aktif apabila suatu kejadian (event) terjadi pada tabel tersebut. Event adalah suatu proses yang terdiri dari INSERT, UPDATE, dan DELETE pada sebuah tabel. Sebagai contoh apabila ada suatu proses query Insert ke tabel 1 maka sebelum (BEFORE) atau sesudah (AFTER) suatu proses query INSERT dijalankan, trigger akan beraksi jika dihubungkan dengan tabel 1 tersebut. Beberapa event yang bisa digunakan untuk mengeksekusi trigger yaitu:

a) Before Insert: dijalankan ketika data dimasukkan dalam tabel
b) After Insert: dijalankan ketika data masuk ke dalam tabel
c) Before Update: dijalankan sebelum proses update data
d) After Update: dijalankan setelah proses update data
e) Before Delete: dijalankan sebelum proses delete data
f) After Delete: dijalankan setelah proses delete data

Sintak dasar penggunaan Trigger seperti terlihat di bawah ini:

```
CREATE TRIGGER nama_trigger [BEFORE|AFTER] [INSERT|UPDATE|DELETE]
ON nama_tabel FOR EACH ROW (statement)
```

Keterangan:

a) Nama_trigger: berisi nama trigger yang akan digunakan
b) Trigger time: kapan eksekusi trigger (Before / After)
c) Trigger event: proses yang terjadi (Insert / Update / Delete)
d) Nama_tabel: berisi nama tabel yang akan di trigger

e) Statement: berisi perintah SQL yang akan diproses. Jika perintah lebih dari satu maka gunakan dalam blok statement BEGIN….END.

10.3.2. After Insert Trigger

Insert trigger merupakan salah satu perintah input data yang terjadi pada suatu tabel dan memicu proses input data pada tabel lain. Perintah trigger ini nantinya juga bisa digunakan sebagai history penambahan data pada sebuah basis data produk. Contoh perintah trigger yaitu:

```
DELIMITER
CREATE TRIGGER hist_barang
AFTER INSERT ON barang FOR EACH ROW
BEGIN
 INSERT INTO loghist_barang SET
 id_barang=loghist_idbarang,
 nama=loghist_nama,
 harga=loghist_harga,
 diskon=loghist_diskon,
 waktu_input=NOW();
END;
DELIMITER;
```

Perintah di atas, menunjukkan bahwa akan terjadi proses input data barang ke dalam tabel loghist_barang setelah terjadi proses input data barang ke dalam tabel barang. Proses tersebut terlihat dari perintah AFTER INSERT pada tabel barang dan selanjutnya terjadi proses INSERT INTO juga pada tabel loghist_barang.

10.3.3. Before Update Trigger

Sama halnya dengan perintah after insert, before update trigger merupakan salah satu perintah ubah data yang terjadi pada suatu tabel dan memicu proses input data pada tabel lain. Perintah trigger ini nantinya juga bisa digunakan sebagai history perubahan data pada sebuah basis data produk. Contoh perintah trigger yaitu:

```
DELIMITER
CREATE TRIGGER histupdate_barang
BEFORE UPDATE ON barang FOR EACH ROW
BEGIN
 INSERT INTO loghist_ubahbarang SET
 id_barang=loghist_idbarang,
 nama=loghist_nama,
 harga=loghist_harga,
 diskon=loghist_diskon,
 waktu_update=NOW();
END;
DELIMITER;
```

Perintah di atas, menunjukkan bahwa akan terjadi proses input data barang ke dalam tabel loghist_ubahbarang setelah terjadi proses input data barang ke dalam tabel barang. Proses tersebut terlihat dari perintah BEFORE UPDATE pada tabel barang dan selanjutnya terjadi proses INSERT INTO juga pada tabel loghist_ubahbarang.

10.3.4. Before Delete Trigger

Before Delete trigger merupakan salah satu perintah input data yang terjadi pada suatu tabel dan memicu proses input data pada tabel lain. Perintah trigger ini nantinya juga bisa digunakan sebagai history penambahan data pada sebuah basis data produk. Contoh perintah trigger yaitu:

```
DELIMITER
CREATE TRIGGER hist_barang
BEFORE DELETE ON barang FOR EACH ROW
BEGIN
 INSERT INTO loghist_hapusbarang SET
 id_barang=loghist_idbarang,
 nama=loghist_nama,
 harga=loghist_harga,
 diskon=loghist_diskon,
 waktu_input=NOW();
END;
DELIMITER;
```

Perintah di atas, menunjukkan bahwa akan terjadi proses input data barang ke dalam tabel loghist_hapusbarang setelah terjadi proses hapus data barang ke dalam tabel barang. Proses tersebut terlihat dari perintah BEFORE DELETE pada tabel barang dan selanjutnya terjadi proses INSERT INTO juga pada tabel loghist_hapusbarang.

10.3.5. Menghapus Trigger

Untuk menghapus trigger yang sudah dibuat, bisa menggunakan perintah:

```
DROP TRIGGER nama_trigger;
```

10.4. SOAL LATIHAN

Dengan menggunakan perintah trigger buatlah tambahan field untuk menyimpan waktu (tanggal) pada proses:

1. pendaftaran anggota pada tabel anggota
2. pendataan buku pada tabel buku

BAB 11. SUBQUERY

11.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami konsep SubQuery pada MySQL.
2) Memahami penggunaan SubQuery pada MySQL.

11.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan konsep SubQuery pada MySQL.
2) Menghasilkan tampilan hasil pengolahan SubQuery pada MySQL.

11.3. URAIAN MATERI

Subquery adalah query didalam query. Artinya seleksi data berdasarkan dari hasil seleksi data yang telah ada. Sintak SQL nya sama dengan sintak SQL pada umumnya, hanya saja kondisi setelah WHERE atau from diikuti dengan query baru atau sub query. Sintak dasar Subquery SQL yaitu:

```
SELECT      field-1, field-2,……, field-n
FROM        nama_tabel
WHERE       kriteria(SELECT field-2,……,
            field-n FROM nama_tabel
            WHERE kriteria);
```

Contoh penggunaan SubQuery seperti terlihat di bawah ini:

```
SELECT * FROM t1 WHERE column1 = (SELECT
column1 FROM t2);
```

Berdasarkan contoh penggunaan SubQuery di atas, **SELECT * FROM t1 ...** merupakan query yang paling luar (outer query), dan **SELECT column1 FROM t2**) merupakan subquery. Bisa dikatakan bahwa subquery adalah sekumpulan query yang ada di outer query, sehingga ada kemungkinan subquery terdapat dalam subquery lagi dan kemudian setumpukan subquery tersebut ada di sebuah query. Subquery bisa diibaratkan anaknya, dan outer query adalah induknya.

Subquery membuat query-query menjadi tersusun, sehingga ada kemungkinan untuk memisahkan / memberi batasan area untuk setiap bagian statement. Subquery menyediakan cara alternatif untuk menjalankan operasi-operasi yang dalam keadaan lain akan membutuhkan JOIN dan UNION yang kompleks. Subquery lebih mudah dibaca dibanding JOIN atau UNION yang kompleks.

Sebuah subquery bisa mengembalikan scalar (single value/satu nilai), single row (satu baris), single column (satu kolom), atau table (satu/lebih baris dari satu/lebih kolom). Ada beberapa batasan untuk tipe statement yang subquery bisa digunakan di dalamnya. Sebuah subquery dengan SELECT yang biasa dapat berisikan: DISTINCT, GROUP BY, ORDER BY, LIMIT, joins, indexhints, UNION, comments, dan fungsi-fungsi lainnya. Sedangkan, outer statement (statement yang paling luar) yang memiliki subquery tidak hanya bisa menggunakan SELECT saja, tapi dia juga bisa menggunakan INSERT, UPDATE, DELETE, SET, atau DO.

11.3.1. Penggunaan Operator Pembanding untuk SubQuery

Penggunaan operator perbandingan di query sudah umum digunakan sesuai dengan kebutuhannya. Operator perbandingan yang umum digunakan antara lain: =, >, <, >=, <=, <>, !=. Perbandingan perintah Subquery juga biasa menggunakan operator LIKE.

11.3.2. Penggunaan SubQuery SELECT Setelah FROM

Subquery dengan statement SELECT pada umumnya digunakan setelah operator perbandingan. Akan tetapi sebenarnya statement SELECT Subquery juga bisa digunakan setelah FROM. Penggunaannya seperti di bawah ini:

```
SELECT . . . FROM (subquery) [AS] name . . .;
```

Tulisan [AS] name adalah wajib dicantumkan, karena setiap label setelah kata FROM harus memiliki nama. Setiap nama kolom di Subquery SELECT harus memiliki nama yang unik. Contoh penggunaan subquery setelah FROM seperti di bawah ini:

```
SELECT sb1, sb2, sb3
FROM (SELECT s1 AS sb1, s2 AS sb2, s3*2 AS sb3 FROM t1) AS sb
WHERE sb1 > 1;
```

Contoh lainnya penggunaan subquery setelah FROM yang ditambahkan dengan fungsi aritmatika akan menjadi seperti di bawah ini:

```
SELECT AVG(sum_column1)
FROM (SELECT SUM(column1) AS sum_column1 FROM t1 GROUP BY column1) AS t1;
```

Sedangkan, subquery (sum_column1) bisa dipanggil oleh outer query yang membutuhkan, contohnya untuk dihitung AVERAGE-nya seperti contoh di atas.

11.3.3. SubQuery 1 Tabel

Subquery bisa digunakan untuk menampilkan data dari 1 tabel maupun multi tabel. Sebagai contoh, perhatikan tabel buku berikut:

KD_BUKU	JUDUL_BUKU	KD_KARANG	KD_TERBIT	JUMLAH
21	Kalkulus	10	1	10
22	Metode Numerik	11	2	20
23	Sistem Basis Data	12	3	40
24	Pengantar Teknologi Informasi	12	3	41
			row(s) 1 - 4 of 4	

Berdasarkan tabel buku di atas, terdapat pertanyaan yaitu Tampilkan judul buku yang jumlahnya paling banyak. Dari pertanyaan tersebut dapat dipahami bahwa diminta untuk menampilkan judul buku yang jumlahnya paling banyak, akan tetapi di pertanyaan tersebut tidak dituliskan kriteria berapa jumlah buku yang paling banyak, maka dengan demikian konsep Subquery yang akan digunakan adalah perlu dicari dulu berapa jumlah terbesar dari jumlah buku yang terdaftar, kemudian baru ditampilkan judul bukunya. Maka dengan demikian perintah Subquerynya adalah:

```
SELECT judul_buku FROM buku
WHERE jumlah = (SELECT MAX(jumlah) FROM buku);
```

11.3.4. SubQuery 2 Tabel

Seperti yang sudah dijelaskan pada subbab sebelumnya bahwa, penggunaan perintah SubQuery bisa digunakan untuk pemrosesan dalam 1 tabel maupun multi tabel. Perhatikan 2 tabel di bawah ini:

Tabel Buku:

KD_BUKU	JUDUL_BUKU	KD_KARANG	KD_TERBIT	JUMLAH
21	Kalkulus	10	1	10
22	Metode Numerik	11	2	20
23	Sistem Basis Data	12	3	40
24	Pengantar Teknologi Informasi	12	3	41
			row(s) 1 - 4 of 4	

Tabel Pengarang:

KD_KARANG	PENGARANG	ALAMAT
12	Adityo PW	Sleman
10	Suparman	Sleman
11	Irwanto	JOgja
	row(s) 1 - 3 of 3	

Berdasarkan 2 tabel di atas, misalkan terdapat pertanyaan untuk menampilkan data menggunakan perintah query, siapa nama pengarang yang jumlah bukunya paling banyak. Sama hal nya dengan pertanyaan sebelumnya, maka perlu dicari dulu jumlah terbanyaknya, kemudian baru direlasikan dengan tabel pengarang untuk menampilkan nama pengarangnya. Sehingga perintah Subquerynya adalah:

```
SELECT pengarang FROM buku, pengarang
WHERE buku.kd_karang=pengarang.kd_karang AND buku.jumlah =
(SELECT MAX(buku.jumlah) FROM buku);
```

11.4. SOAL LATIHAN

Buatlah tabel dengan komposisi seperti dibawah ini:

Buku(kd_buku, judul_buku, pengarang, penerbit, jml_buku)

Petugas(kd_petugas, nama_petugas, alamat_petugas)

Pelanggan(kd_pelanggan, nama_pelanggan, alamat_pelanggan)

Pembelian(kd_beli, kd_petugas, kd_pelanggan, tanggaljual, total_jual, total_item)

DetailPembelian(kd_detail, kd_beli, kd_buku, jml_beli)

Berdasarkan komposisi tabel diatas, jawablah pertanyaan berikut menggunakan perintah query.

1. Tampilkan nama pelanggan yang melakukan pembelian dengan total item terbanyak.
2. Tampilkan nama pelanggan yang mempunyai alamat yang sama.
3. Tampilkan judul buku yang dibeli paling banyak
4. Tampilkan judul buku yang jumlahnya diatas rata-rata

BAB 12. QUERY DCL (DATA CONTROL LANGUAGE)

12.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami pengertian dan fungsi user pada sebuah database.
2) Memahami pengertian dan fungsi privileges pada sebuah database.
3) Memahami cara pengaturan manajemen user, privileges, grant, dan revoke.

12.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Membuat user pada sebuah database.
2) Melakukan pengaturan privileges user pada sebuah database.
3) Melakukan manajemen user dan privileges menggunakan perintah grant, dan revoke pada sebuah database.

12.3. URAIAN MATERI

Pengguna basis data atau database user merupakan hal yang sangat penting dalam sebuah pengelolaan basis data. pada umumnya database user terdiri dari Database Administrator (DBA) dan end user. DBA merupakan pengguna yang paling utama dimana mempunyai hak akses paling banyak bahkan hampir semua hak akses pada database dimiliki oleh DBA. Hak akses yang dimaksud adalah manipulasi database dan data didalam database tersebut, contohnya antara lain create database, create table, alter table, drop table, insert data dalam table, update data dalam table, delete data dalam table dan lain sebagainya. Pengguna lain selain DBA adalah end user. End User hanya bisa mengakses dan mengelola database berdasarkan akses yang diberikan oleh DBA. Pemberian hak akses pada databse menggunakan perintah grant, sedangkan untuk membatalkan hak akses yang diberikan menggunakan perintah revoke. Dalam istilah query, perintah untuk mengatur pengguna basis data dinamakan Data Control Language (DCL).

12.3.1. User

Setiap DBMS menyediakan satu akun default yang nantinya bisa digunakan sebagai DBA. Pada DBMS MySQL, akun default yang biasa digunakan untuk DBA login yaitu menggunakan username root dengan password kosong (tanpa password). Jika salah satu sudah ada yang bertindak sebagai DBA, maka untuk alasan keamanan sebaiknya password user root segera diganti dan dibuatkan pengguna baru dibawah root.

Dalam aplikasi basis data, khususnya yang berhubungan dengan aplikasi komputer, hak akses terkait operasi query jumlahnya sedikit dan tidak dilakukan secara bersama-sama. Oleh karena itu setiap perintah basis data, tidak perlu semuanya menggunakan user dengan level root.

Perintah dasar membuat user pada basis data khususnya MySQL, seperti terlihat pada sintak di bawah ini:

```
CREATE USER nama_user@nama_mesin IDENTIFIED by password
```

Keterangan sintak:

- Nama_user : nama user yang akan dibuat
- Nama_mesin : asal mesin atau host/IP Address user mengakses.
  Jika diakses menggunakan host/komputer itu sendiri (lokal), maka diisi "localhost". Namun jika diakses dari mesin lain bisa menggunakan "%".
- Password : password yang digunakan.

Contoh 1: Penggunaan sintak *Localhost*:

```
CREATE USER admin@localhost IDENTIFIED by Adm1n321
```

Berdasarkan perintah diatas, artinya adalah membuat user dengan nama "Admin" dari mesin "Localhost" dan dengan password "Adm1n321".

Contoh 2: Penggunaan sintak pada mesin lain:

```
CREATE USER admin@% IDENTIFIED by Adm1n321
```

Berdasarkan perintah diatas, artinya adalah membuat user dengan nama "Admin" dari mesin atau komputer manapun dan dengan password "Adm1n321". Tanda '%' mirip seperti LIKE Statement pada perintah SQL.

Jika ingin menampilkan perintah user yang sudah dibuat dengan perintah diatas, bisa menggunakan perintah di bawah ini:

```
SELECT User, Password, Host FROM user
```

User yang sudah dibuat juga bisa dihapus, dengan menggunakan perintah di bawah ini:

```
DROP user nama_user
```

12.3.2. Privilege

Privilege adalah hak akses atas sesuatu, sedangkan System Privilege adalah hak akses terhadap database. Hak akses untuk memanipulasi isi database objek adalah object privilege. Perintah yang digunakan untuk memberiakn hak akses database terdiri dari:

a) Select_priv : memberikan hak user untuk menampilkan data.
b) Insert_priv : memberikan hak user untuk memasukan data.
c) Update_priv : memberikan hak user untuk merubah/memperbarui data
d) Delete_priv : memberikan hak user untuk menghapus data
e) Drop_priv : memberikan hak user untuk menghapus database, tabel, kolom.
f) Reload_priv : memberikan hak user untuk melakukan refresh server.
g) Shutdown_prive : memberikan hak akses user untuk mematikan daemon server MySQL.
h) Process_priv : memberikan hak user untuk melihat proses server MySQL.
i) File_priv : memberikan hak user untuk mengkonversi data dari database menjadi sebuah file.
j) Grant_priv : memberikan hak user untuk memberikan hak akses ke user lain.
k) References_priv :
l) Index_priv : memberikan hak user untuk membuat dan menghapus index pada database.
m) Alter_priv : memberikan hak user untuk merubah objek database.
n) Show_db_priv : memberikan hak user untuk melihat semua database.
o) Super_priv : memberikan hak user untuk melakukan pengaturan pada database.
p) Create_tmp_table_priv : memberikan hak user untuk membuat temporary tabel.
q) Lock_tables_priv : memberikan hak user untuk memberikan kunci (password) untuk mengakses suatu tabel.
r) Repl_slave_priv : memberikan hak user untuk membaca logs pada master replikasi.

12.3.3. Grant

Grant merupakan salah satu perintah SQL yang termasuk ke dalam kelompok DCL (Data Contor Language). Perintah grant digunakan untuk memberikan/mengijinkan user untuk mengakses tabel di dalam sebuah database. Terdapat beberapa perintah grant yang biasa digunakan sesuai dengan kebutuhannya. Perintah Grant hanya bisa dilakukan oleh administrator pada database tersebut.

Beberapa perintah dasar grant, yaitu:

a) CREATE: memperbolehkan user membuat basis data/tabel
b) SELECT: memperbolehkan user melakukan pencarian data/menerima data
c) INSERT: memperbolehkan user menambah data baru pada sebuah tabel
d) UPDATE: memperbolehkan user merubah data baru pada sebuah tabel
e) DELETE: memperbolehkan user menghapus data baru pada sebuah tabel
f) DROP: memperbolehkan user menghapus seluruh basis data/tabel

Pemakaian perintah diatas menggunakan perintah SQL. Sintak SQL perintah grant seperti terlihat di bawah ini.

a) Sintak Umum

```
GRANT hak_akses ON nama_database.nama_tabel TO
                user@nama_host
```

b) Sintak pengaturan untuk hak akses penuh kepada suatu user ‘tyo’ dan pada host ‘localhost’

```
GRANT ALL PRIVILEGES ON *.* TO 'tyo'@'localhost'
```

c) Sintak pengaturan untuk hak akses tertentu (select, insert) pada user ‘tyo’ dan pada server ‘localhost’

```
GRANT select,insert ON *.* TO 'tyo'@'localhost'
```

12.3.4. Revoke

Sama halnya dengan Grant, Revoke juga merupakan salah satu perintah SQL yang termasuk ke dalam kelompok DCL (Data Contor Language). Kebalikan dengan perintah grant, revoke digunakan untuk membatalkan ijin user yang sudah diberikan dari perintah grant untuk mengakses tabel di dalam sebuah database. Penggunaan perintah revoke yang biasa digunakan seperti di bawah ini:

a) Sintak umum

```
REVOKE hak_akses ON nama_database.nama_tabel FROM
                user@nama_host
```

b) Sintak menghapus hak akses penuh yang sudah diberikan kepada user ‘tyo’

```
REVOKE ALL PRIVILEGES ON *.* FROM 'tyo'@'localhost'
```

12.4. LATIHAN

12.4.1. Membuat user baru

Pembuatan user baru di database MySQL pada dasarnya bisa menggunakan perintah query insert, dimana akan dimasukan data user baru ke dalam tabel user. Tabel user merupakan tabel default dari MySQL yang menyimpan semua data-data user yang akan mengakses database MySQL. Perintah untuk membuat user baru ke dalam tabel user seperti di bawah ini:

```
CREATE USER tyo@localhost IDENTIFIED by ty0321
```

Perintah diatas menunjukkan bahwa membuat user baru dengan nama 'tyo' yang mengakses database secara '*localhost*', password loginnya adalah 'ty0321'. Setelah perintah menambahkan user sudah berhasil dieksekusi, selanjutnya gunakan perintah '***Flush Privileges'***. Perintah ini digunakan untuk me-reload, memerintahkan server untuk membaca ulang hak akses.

12.4.2. Memberikan hak akses user

Jika user sudah berhasil dibuat, selanjutnya diberikan wewenang/hak akses kepada user baru tersebut. Pemberian hak akses menggunakan perintah query update. Contoh sintaknya seperti di bawah ini:

```
UPDATE user SET select_priv='y',
        insert_priv='y',
        update_priv='y',
        delete_priv='y',
        create_priv='y',
         drop_priv='y',
         alter_priv='y'
       WHERE user='tyo';
```

Berdasarkan perintah query diatas, menunjukkan bahwa diberikan hak akses yang terdiri dari pencarian data (select_priv), menambahkan data (insert_priv), merubah data (update_priv), hapus data (delete_priv), membuat tabel (create_priv), hapus tabel (drop_priv), dan rubah struktur tabel (alter_priv).

12.5. TUGAS / PR

1. Buatlah beberapa user dengan hak akses berbeda-beda sesuai dengan jenis-jenis hak akses dalam basis data. Ketentuan hak akses terdiri dari:
   a. User hanya bisa melihat data
   b. User bisa melihat dan menambah data
   c. User hanya bisa memperbaiki data
   d. User bisa memperbaiki dan menghapus data
   e. Use rmemiliki semua hak akses
2. Uji masing-masing user dengan koneksi ke MySQL menggunakan SQLYog. Pastikan user dengan privileges dapat melihat data, tidak bisa melakukan penambahan, perbaikan dan pengapusan data.

BAB 13. NoSQL dengan FIREBASE

13.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Mampu memahami konsep NoSQL
2) Mampu memahami penggunaan Firebase untuk pengelolaan basis data

13.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menjelaskan dan mengimplementasikan basis data dengan konsep NoSQL
2) Mengelola basis data menggunakan Firebase

13.3. URAIAN MATERI

Basis data dengan konsep NoSQL berbeda dengan basis data relasional seperti MySQL. Basis Data NoSQL artinya adalah Not-Only SQL, maksudnya menyatakan bahwa database masih memiliki dukungan atau mengenali yang mirip seperti bahasa query, namun menggunakan format penulisan yang berbeda.

Firebase merupakan salah satu Realtime Database yang menggunakan konsep NoSQL. Firebase menggunakan Format JSON yang digunakan untuk menyimpan, membaca, menulis dan mengambil data secara Realtime yang disingkronkan kepada client yang terhubung. Dengan menggunakan Firebase, walaupun klien tidak terkoneksi internet, client dapat menerima setiap perubahan data yang ada, saat client terkoneksi kembali dengan Internet, peristiwa realtime akan terus berlangsung, sehinggal user dapat pengalaman yang responsif.

Beberapa perbedaan istilah yang ada di RDBMS dan NoSQL, antara lain:

No.	RDBMS	NoSQL
1	Database	Database
2	Tabel	Collection
3	Baris	Document atau BSOM Document
4	Kolom	Field
5	Indeks	Index
6	Table Join	Embedded Document dan Link
7	Primary Key	Primary Key

Firebase merupakan salah satu layanan dari Google yang biasa dikenal dengan Firebase Google yaitu Backend as a Service untuk membuat aplikasi mobile maupun web. Produk dari Firebase Google sendiri saat ini terdapat dua produk yakni, pertama adalah Firebase Realtime Database, kedua adalah Remote Config.

Keuntungan firebase:

1) Terdapat versi free yang dapat diakses secara unlimited.
2) Dapat melakukan sinkronisasi data menggunakan Realtime Database.
3) Dapat diakses secara offline dan aplikasi ini tetap responsive saat offline karena adanya SDK Firebase Realtime.
4) User friendly.
5) Bisa diakses melalui seluler atau browser.

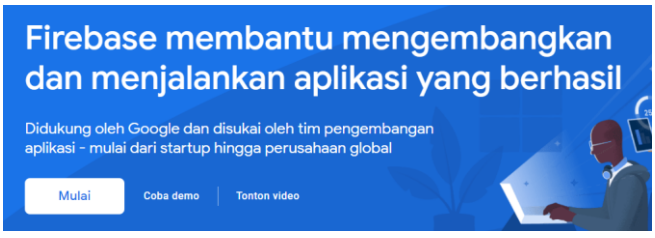
CATATAN, beberapa hal yang perlu diperhatikan dan dipersiapkan sebelum membuat database menggunakan Firebase.

1) Pastikan Android Studio sudah dalam versi yang paling terbaru
2) API target level 16 (jellybean) atau yang paling baru.
3) Menggunakan Gradle 4.1.
4) Gunakan Jetpack (AndroidX) yang telah mencakup: com.android.tools.build:gradle v3.2.1 atau yang terbaru
5) CompileSdkVersion 28 atau yang terbaru.
6) Siapkan perangkat fisik atau jalankan emulator untuk menjalankan emulator
7) Masuk kedalam akun Firebase kamu

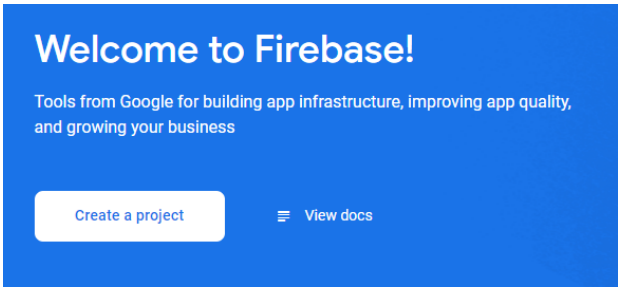
13.3.1. Membuat Database di Firebase

Untuk membuat database di dalam firebase, sebelumnya perlu registrasi atau mendaftar di firebase.google.com. setelah melakukan registrasi, ikuti Langkah-langkah di bawah ini.

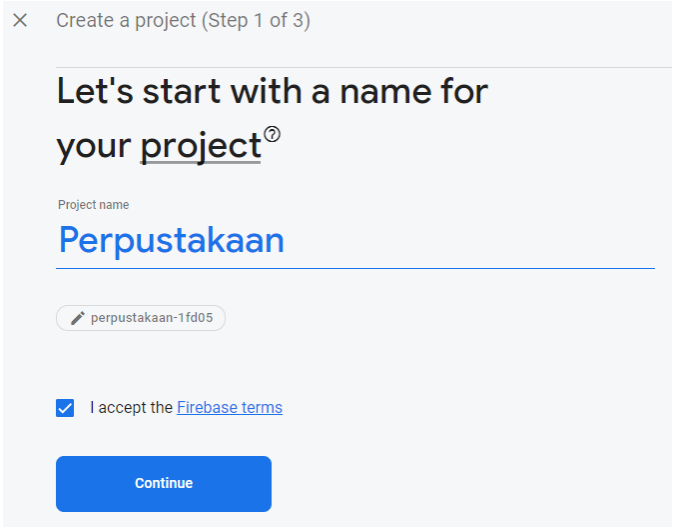
1) Klik Mulai untuk memulai membuat proyek firebase.



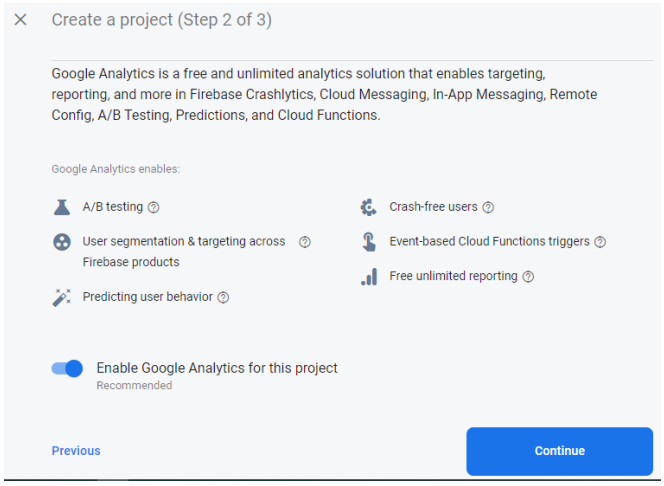
2) Klik Create Project



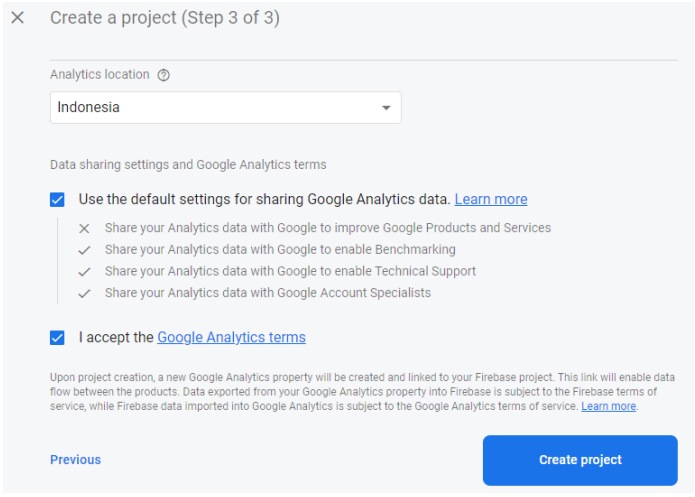
3) Kemudian isikan isikan Project Name, misalkan Namanya Perpustakaan



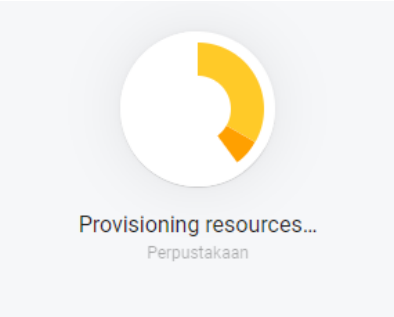
4) Klik Continue pada halaman Google Analytics



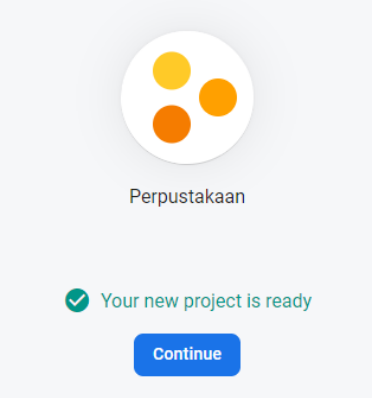
5) Pada tampilan Analytic Location pilih Indonesia, kemudian beri checklist pada ketentuan Google Analytic Terms.



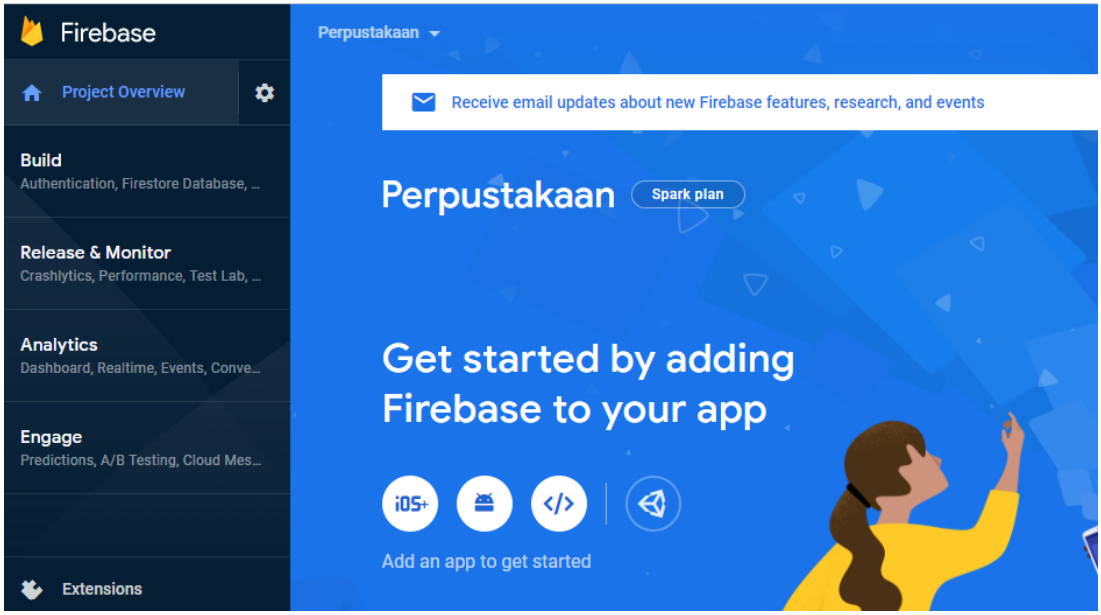
6) Tunggu Proses generate yang dilakukan oleh google



7) Proyek sudah ready digunakan

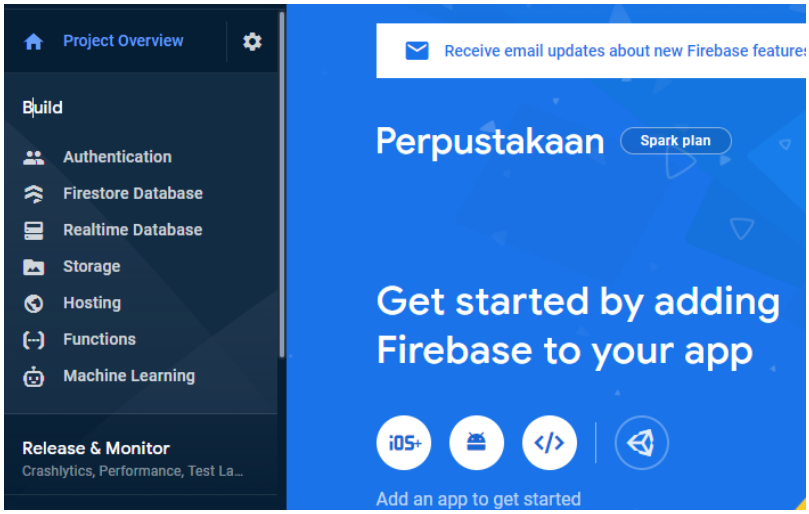


8) Pilih jenis mobile yang akan kita gunakan untuk membuat proyek tersebut. Misalkan ada iOS atau Android.

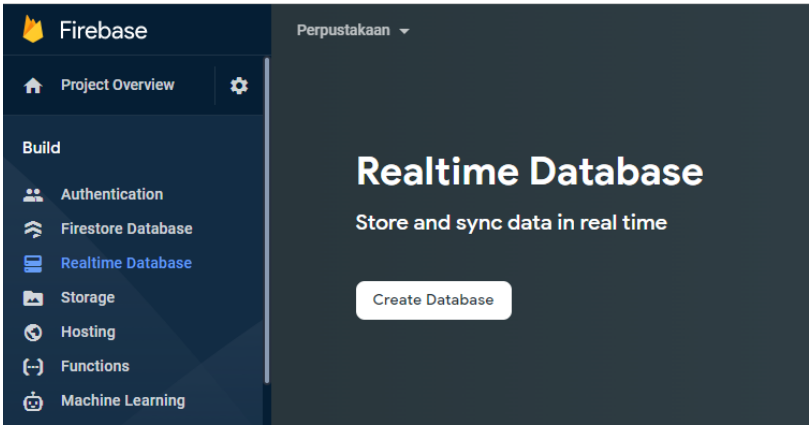


13.3.2. Mulai membuat Database di Firebase

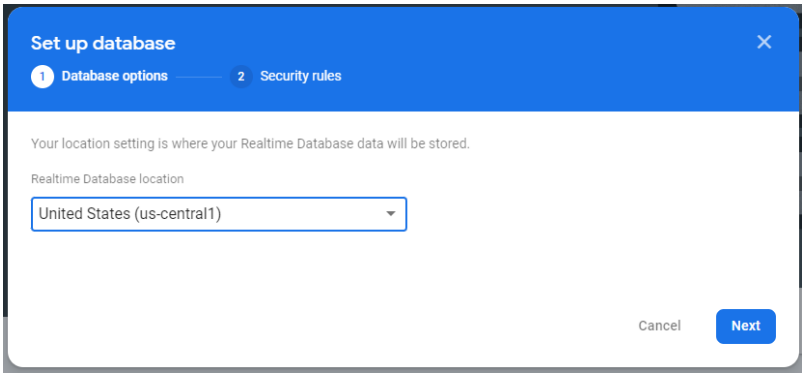
Pada tampilan poin nomor 8 di atas, lihat di bagian menu sebelah kiri, klik Build kemudian pilih Realtime Database, seperti tampilan di bawah ini.



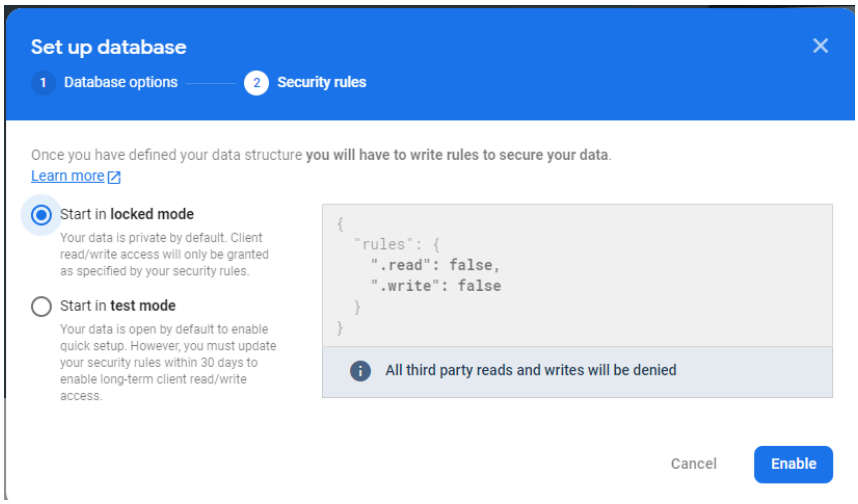
Selanjutnya adalah mulai membuat Database dengan klik tombol Create Database seperti pada gambar di bawah ini.



Langkah pertama saat Setup database, isikan Realtime Database Location. Hanya ada 3 pilihan, maka silakan dipilih United States, seperti pada gambar di bawah ini.



Langkah ke dua setup database, pilih Start in Locked Mode, seperti gambar di bawah ini.



Database yang sudah siap untuk digunakan akan tampil seperti di bawah ini.

Selanjutnya yang perlu dilakukan adalah memastikan bahwa sudah mulai membuat aplikasi di Androidnya, karena dari Firebase perlu langsung dikoneksikan dengan aplikasi android yang dibuat. Setelah dikoneksikan dengan Firebase, aplikasi android mulai bisa digunakan sebagaimana mestinya.

13.4. TUGAS / PR

Buatlah salah satu aplikasi sederhana di Android (minimal 1 form untuk mengakses 1 penyimpanan). Kemudian buatlah databasenya menggunakan Firebase sampai bisa dilakukan proses CRUD menggunakan aplikasi android tersebut.

BAB 14. TUGAS AKHIR PERANCANGAN BASIS DATA

14.1. KOMPETENSI DASAR

Setelah mempelajari bab ini, mahasiswa:

1) Memahami konsep perancangan basis data mulai dari penentuan entitas sampai dengan perancangan relasi tabel.
2) Memahami cara implementasi basis data dari perancangan yang sudah dibuat menggunakan sebuah aplikasi DBMS MySQL.
3) Memahami konsep implementasi basis data menggunakan NoSQL.

14.2. INDIKATOR

Setelah mempelajari bab ini, mahasiswa mampu:

1) Menghasilkan rancangan basis data mulai dari penentuan entitas, penggambaran ER-Diagram sampai dengan relasi tabel
2) Menghasilkan basis data menggunakan DBMS MySQL.
3) Menghasilkan basis data menggunakan konsep NoSQL.

14.3. URAIAN MATERI

Seorang pengusaha perusahaan penjualan barang spare part motor mengalami permasalahan terkait pendataan barang jualannya. Pengusaha tersebut membutuhkan aplikasi yang bisa mendata stok barang dan keuntungan penjualan. Pengguna aplikasi tersebut adalah petugas Gudang, petugas kasir dan admin (pemilik). Stok barang bertambah dengan adanya transaksi pembelian barang dari supplier yang didata dan dilayani oleh petugas Gudang. Sementara stok berkurang dari adanya transaksi pembelian barang oleh konsumen (member) yang dilayani oleh kasir. Admin bisa memantau stok barang dan keuntungan yang didapat selama periode tertentu.

14.4. TUGAS/PR

Berdasarkan uraian di atas, Anda sebagai seorang *database analyst*, diminta untuk:

1) Membuatkan sebuah rancangan basis data untuk membantu permasalahan pengusaha tersebut.
2) Implementasikan rancangan pada poin 1 di DBMS MySQL.
3) Dikarenakan kedepannya si pengusaha berencana akan mengembangkan aplikasinya menggunakan aplikasi mobile, buatkan pula database untuk mobilenya menggunakan firebase.

diterbitkan oleh :
Universitas Teknologi Yogyakarta
Jl. Siliwangi, Jombor, Sleman, Yogyakarta
Email : publikasi@uty.ac.id
Website : uty.ac.id