RANCANG BANGUN DAN OPTIMASI BASIS DATA SISWA PADA LEMBAGA KURSUS BAHASA PRIBADI

ABSTRAK

Tujuan penelitian ini adalah untuk merancang basis data siswa di Lembaga Kursus Pribadi yang dapat menyediakan struktur informasi yang mudah dimengerti dan mendukung kebutuhan pemrosesan administrasi siswa di lembaga tersebut. Perancangan secara garis besar meliputi perancangan konseptual, perancangan logical, dan perancangan fisik dan alat bantu antara lain data flow diagram, entity relationship diagram dan teknik normalisasi. Dari penelitian ini akan dihasilkan rancang bangun basis data yang optimal dan terorganisir yang dapat memenuhi kebutuhan organisasi.

Kata Kunci: basis data, perancangan, siswa

Baby Lolita B.

b_lolita@staff.gunadarma.ac.id

PENDAHULUAN

Seiring dengan meningkatnya kompleksitas data, Pribadi sebagai lembaga yang bergerak di bidang pendidikan, membutuhkan sistem pengolahan data yang lebih baik dan terorganisir. Dalam perkembangannya organisasi ini mengalami pemekaran sehingga dibuka cabang-cabang baru di lokasi yang tersebar. Untuk itu perlu dilakukan perancangan basis data kesiswaan di lembaga kursus Pribadi sesuai kebutuhannya.

METODE PENELITIAN

Penelitian ini meliputi enam tahap proses perancangan basis data yaitu :

1. Pengumpulan data dan analisis, dengan menggunakan *tools DFD*.
2. Perancangan basis data secara konseptual, dengan menggunakan *tools ERD*.
3. Pemilihan DBMS
4. Perancangan basis data secara logika, dengan menggunakan *tools* normalisasi.
5. Perancangan basis data secara fisik.
6. Implementasi sistem basis data

HASIL PEMBAHASAN

Dalam melaksanakan proses-proses bisnis, supaya dapat berjalan secara baik dan teratur, maka di lembaga ini diterapkan Standard Operating Procedure (SOP) untuk bagian kesiswaan. SOP terdiri dari SOP penerimaan siswa, SOP pembayaran, SOP siswa pindah cabang, dan SOP organisasi data.

SOP Penerimaan Siswa (Pendaftaran dan *Placement Test*) meliputi (a) siswa mengisi formulir yang telah disediakan. Ini bermanfaat bagi siswa lama untuk *updating data*. (b). bagian pendaftaran. Bagian ini juga bertugas sebagai bagian penyimpanan data, dan sebab itu akan menyimpan data siswa baru atau *update* data siswa lama ke dalam *spreadsheet*. (c) bagi siswa baru akan dilakukan tes penempatan untuk menentukan level kelas yang dapat ditempati. Bagi siswa yang sudah terdaftar sebelumnya (siswa lama) tidak dilakukan tes penempatan, Ini tidak berlaku bagi siswa lama yang telah mengundurkan diri selama lebih dari satu tahun dan hendak bergabung kembali.

SOP pembayaran meliputi (a) siswa melakukan pembayaran ketika mengisi form pendaftaran; (b) pembayaran dapat dilakukan secara tunai ataupun dicicil tiga kali dengan jumlah cicilan yang ditentukan oleh lembaga; (c) pembayaran yang dilakukan secara tunai akan mendapatkan diskon.

SOP siswa pindah cabang meliputi (a) siswa yang hendak pindah ke cabang lain harus meminta izin (melaporkan diri) ke cabang di mana ia sekarang berada; (b) kantor cabang yang lama akan mencarikan kelas dengan level yang sama pada cabang yang diinginkan oleh siswa yang ingin pindah; (c) pada cabang baru, siswa yang baru pindah hanya perlu mengisi formulir pendaftaran saja dan tidak melakukan registrasi kembali.

SOP organisasi data adalah sebagai berikut (a) data-data yang diperlukan (telah distandarisasi dalam sistem informasinya yang berbasis Excel) harus disimpan ke dalam *hard disk*; (b) secara berkala data yang ada dalam hard disk di*back up* ke dalam dua disket. Disket pertama akan disimpan oleh kantor cabang sedang disket kedua diberikan ke kantor pusat; (c) kantor cabang harus memberikan data yang diperlukan oleh kantor pusat (*back up* data yang ada di *hard disk* kantor cabang) secara berkala.

Pengumpulan Data dan Analisis

Pada tahapan ini digunakan DFD untuk mengumpulkan dan menganalisis data yang dibutuhkan dalam pengembangan sistem Kesiswaan., DFD dibuat hingga tiga tingkatan.

Pada sistem kesiswaan seperti yang terlihat pada Gambar 1, terdapat dua terminator, yaitu *administrator* pusat dan *administrator* cabang. Kedua *administrator* ini dipisah karena ada peran-peran yang berbeda antara *administrator* cabang dan pusat.

Diagram Zero

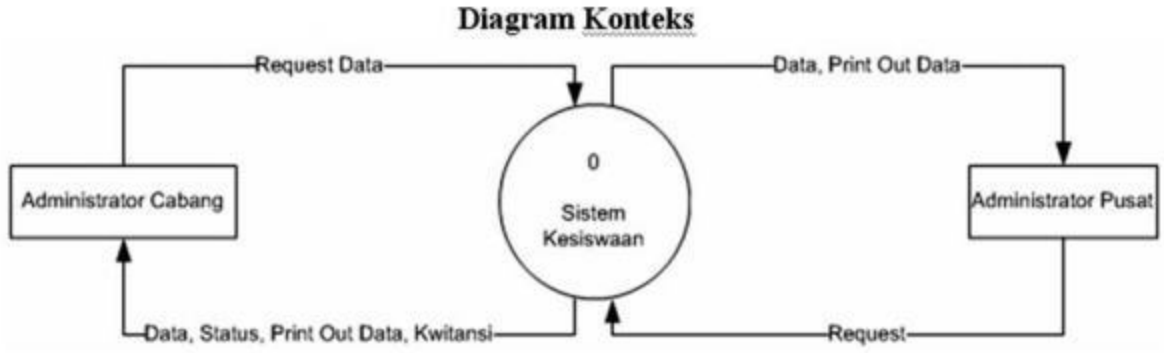
DFD zero dapat dilihat pada Gambar 2 di mana sistem kesiswaan terdiri dari 3 sub-sistem yaitu sub-sistem Akademik, Registrasi, dan Login. Untuk sub-sistem Akademik dan Registrasi akan dijelaskan lebih lanjut secara lebih mendetail di halaman-halaman selanjutnya.

Untuk sub-sistem login, administrator harus memasukkan data login mereka berupa *username*, *password*, dan cabang tempat *administrator* tersebut bekerja. Data ini merupakan syarat bagi administrator agar dapat mengunakan sistem. Setiap kali *user* melakukan login, maka *user* akan mendapatkan status keberhasilan atau tidaknya login yang dilakukannya.

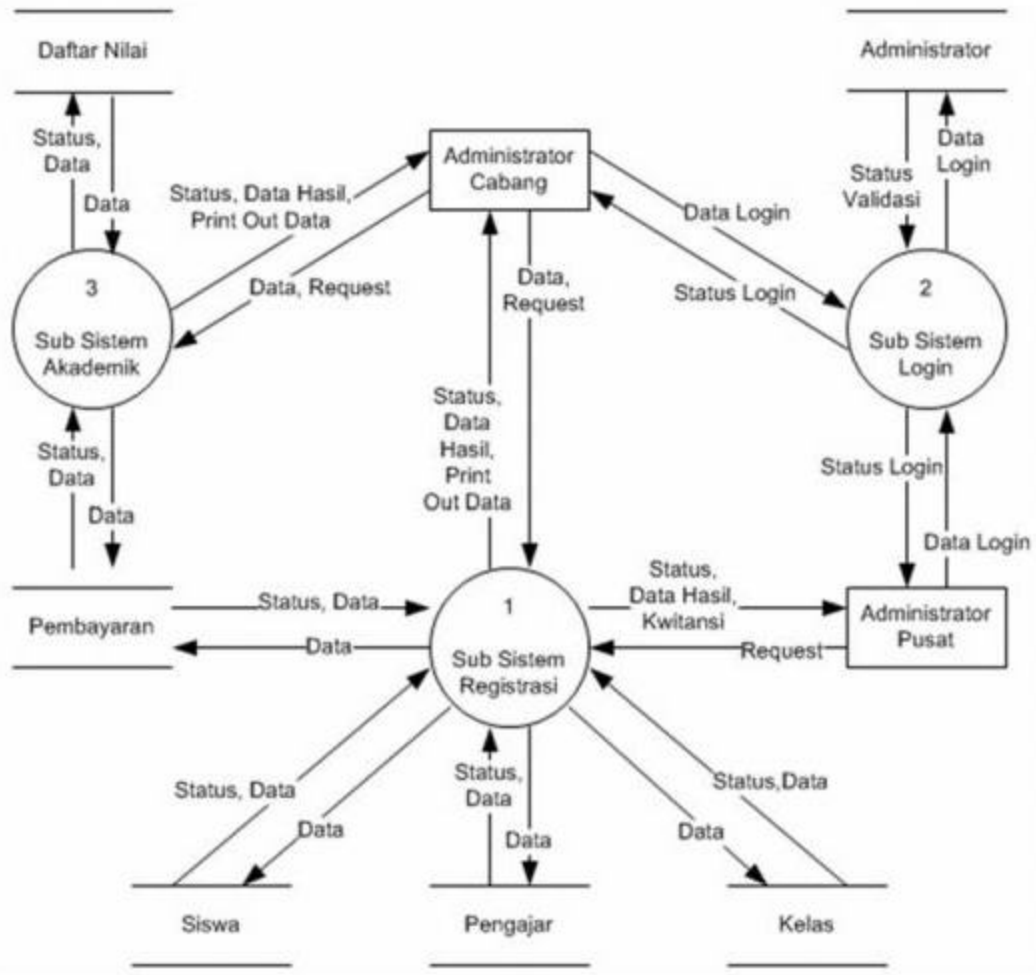
Diagram Detail Sub-sistem Registrasi

Pada subsistem registrasi (Gambar 3), *user* dapat melakukan fungsi pencatatan pembayaran, perubahan ataupun penambahan data siswa, kelas, dan pengajar. Sistem ini juga memungkinkan *user* mengatasi masalah pergantian cabang. Setiap kali *user* melakukan *request* atau penambahan data dan sebagainya, *user* akan mendapatkan *status report* berhasil atau tidak.

Ketika ada siswa yang melakukan pembayaran (secara tunai maupun



Gambar 1. Diagram Konteks



Gambar 2. Diagram Zero

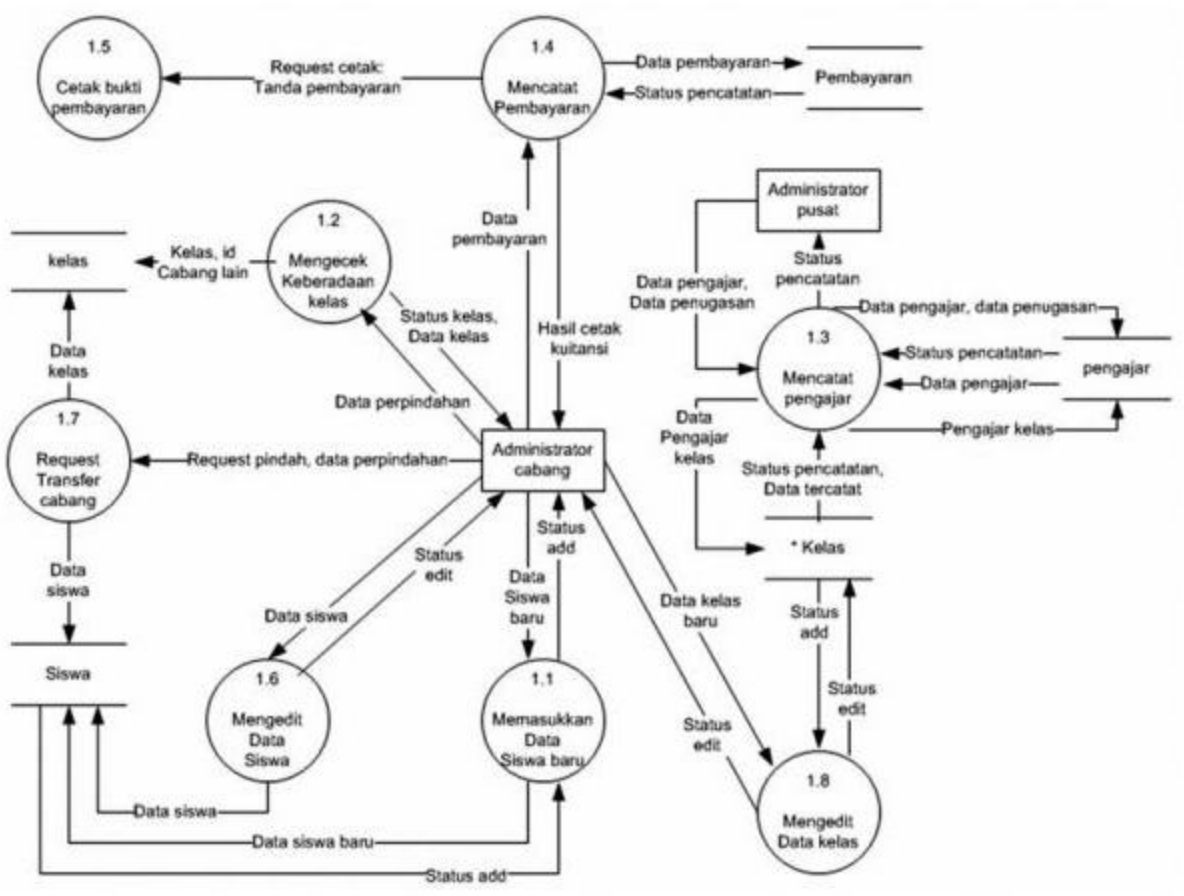
cicilan), maka administrator cabang akan memasukkan data pembayaran (nama siswa, level, kelas, jenis pembayaran, dll) pada sistem. Selanjutnya data tersebut diproses untuk dicatat. Data-data tersebut dimasukkan ke dalam tabel pembayaran. Selanjutnya apabila pencatatan telah dilakukan, maka secara otomatis bukti pembayaran yang berupa kuitansi (1.5) yang kemudian akan diberikan oleh *administrator* kepada siswa tersebut.

Dalam pencatatan data siswa, apabila calon siswa yang sudah mengetahui hasil *placement test*-nya, kemudian mengembalikan formulir registrasi ulang dan juga melakukan pembayaran, maka administrator cabang akan mencatat data calon siswa tersebut (yang ada pada *form* biodata yang telah diisinya) sebagai siswa (1.1).

Apabila ada perubahan pada data kelas, baik karena pergantian pengajar, siswa yang tidak mendaftar ulang lagi, dan lain-lain, maka data-data perubahan yang ada, misalnya jumlah siswa dan lain sebagainya, menjadi bahan untuk mengedit *database* kelas.

Apabila ada perubahan pada data siswa, karena siswa pindah rumah atau mengganti nomor telepon, dan lain-lain, maka data perubahan yang ada menjadi bahan untuk mengedit *database* siswa. Proses mencatat pengajar kelas yang telah dibuat sebelumnya pada DFD, pada DFD ini dipisahkan menjadi dua proses dikarenakan perbedaan-perbedaan yang ada.

Proses yang pertama dilakukan oleh administrator pusat, sedangkan yang kedua dilakukan oleh administrator cabang. Ketika ada penambahan pengajar atau pemindahan tugas pengajar untuk mengajar pada suatu cabang (umumnya pada awal *term*), administrator akan memasukkan data pengajar dan data



Gambar 3 Diagram Detail Sub-sistem

penugasan melalui *client* untuk dimasukkan oleh *server* ke dalam tabel Pengajar (1.3). Apabila pencatatan telah dilakukan, maka *database* akan mengirimkan *report* berupa status keberhasilan pencatatan kepada *client* untuk diberikan kepada administrator melalui *interface*.

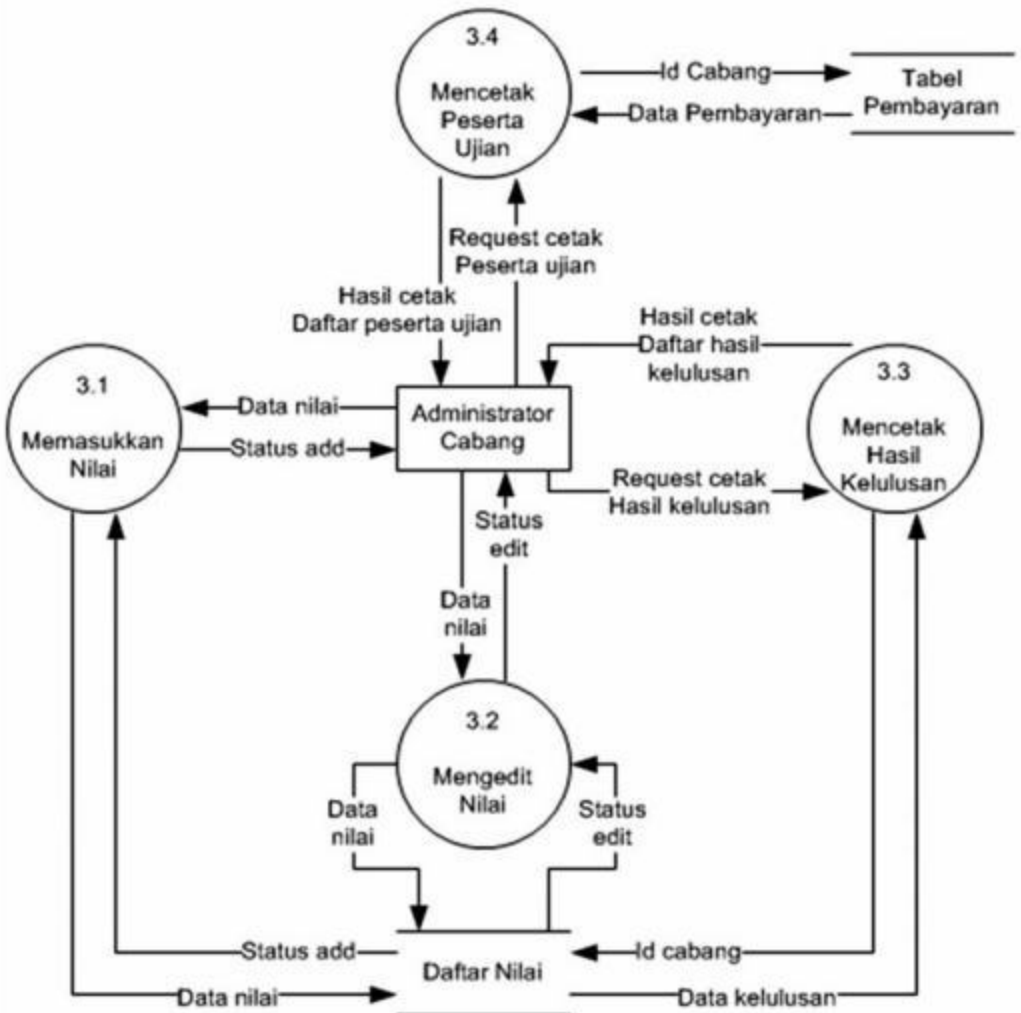
Pada saat awal *term* terjadi (waktu di mana suatu kelas mungkin mengalami pergantian pengajar), administrator cabang akan menghubungi pusat sehubungan dengan pengajar-pengajar yang dibutuhkan di cabang (dikerjakan di luar sistem). Administrator cabang dapat melakukan proses transfer siswa antar cabang dengan cara memasukkan *request transfer*. Proses transfer cabang akan meminta *input* berupa data siswa dan kelas yang ingin melakukan transfer.

Sebelum *user* melakukan transfer siswa dari cabang satu ke cabang yang lain, *user* harus bisa mendapatkan kelas yang sesuai dengan siswa. Untuk melakukan pencarian kelas-kelas yang ada di setiap cabang dan pusat, administrator cabang dapat melakukan request mengecek keberadaan kelas. Request dilakukan dengan memasukkan *input* berupa data perpindahan. *Input* data perpindahan yang diperlukan mencakup kelas yang ingin dicari dan Id cabang. Kemudian dilakukan *query* terhadap database kelas untuk mencari kelas yang diinginkan *user*. Setelah *user* menerima daftar kelas yang sesuai dengan keinginannya, baru kemudian user bisa melanjukan kembali proses *transfer* cabang.

Dalam sub-sistem akademik seperti terlihat pada Gambar 4, *user* yang dapat melakukan setiap proses adalah administrator cabang. Proses-proses yang dapat dilakukan meliputi tambah nilai siswa, *edit* nilai siswa, *print* peserta ujian, dan *print* hasil kelulusan. *User* dapat melakukan alur proses tambah nilai dengan memasukkan *input* berupa data nilai, untuk kemudian ditambahkan data baru tersebut ke dalam *database* Daftar nilai. *User* juga dapat melakukan alur proses *edit* nilai dengan memasukkan *input* berupa data apa yang ingin dilakukan pengeditan kemudian akan dilakukan *update* ke *database* Daftar Nilai.

User melakukan alur proses *print* peserta ujian dengan cara melakukan *request print* peserta ujian kemudian akan dicek siswa yang sudah melunasi pembayarannya. Selanjutnya akan dihasilkan daftar siswa yang diperbolehkan untuk mengikuti ujian karena sudah melunasi kewajiban pembayaran. Dari data ini, kemudian akan dilakukan *request* ke *device printer*

Diagram Detail Subsistem Akademik



Gambar 4. Diagram Detail Sub-sistem Akademik

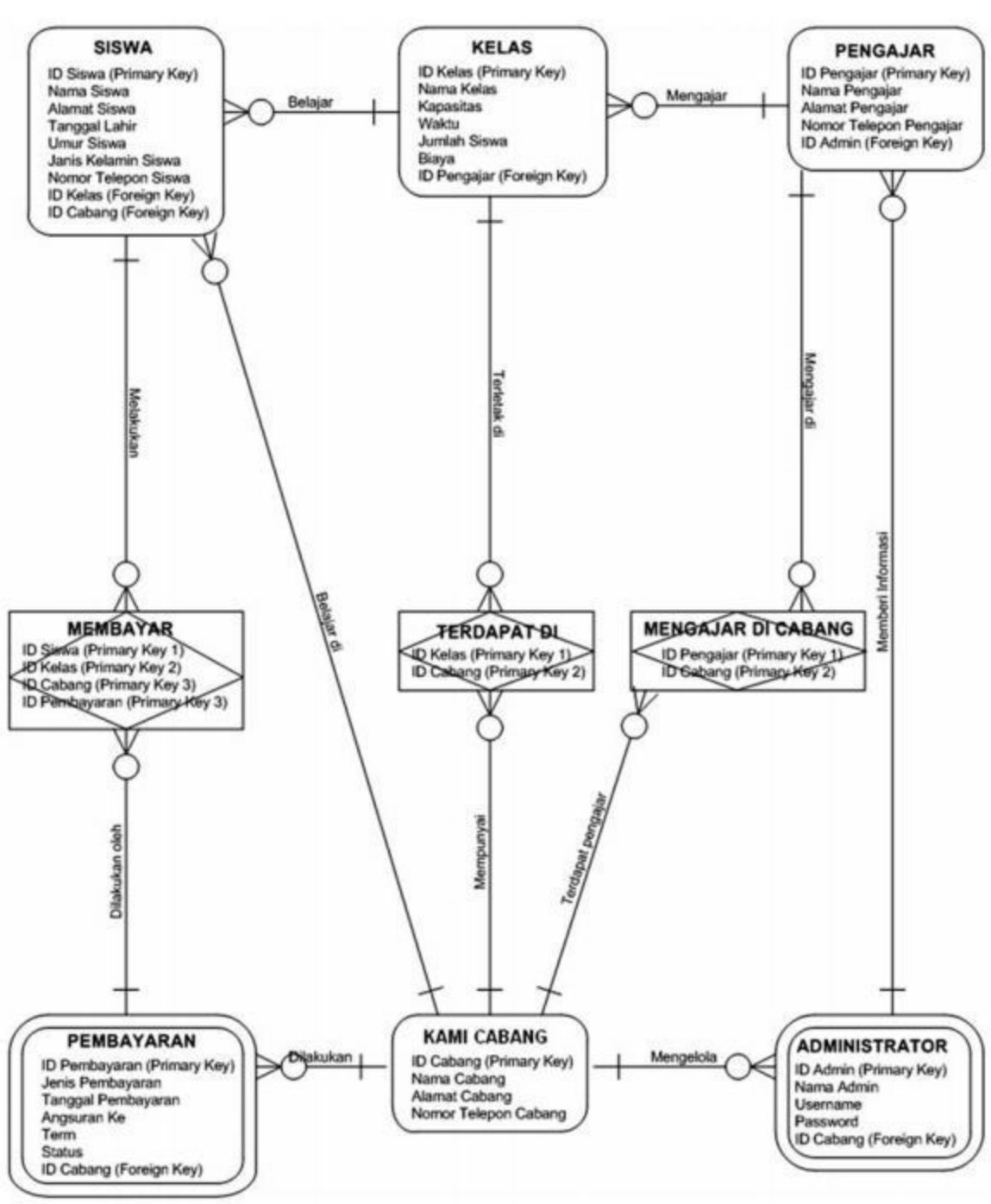
untuk melakukan pencetakan terhadap data daftar siswa peserta ujian.

User melakukan alur proses *print* hasil kelulusan dengan cara memasukkan *request* ke sistem untuk mencetak hasil kelulusan. Kemudian akan dilakukan verifikasi nilai ke dalam *database* Daftar Nilai, dan selanjutnya hasilnya akan dicetak.

Perancangan Basis Data Secara Konseptual

Gambar 5 memperlihatkan *Entity Relationship Diagram* (ERD) dari sistem yang dikembangkan. Entitas dan relasi yang ada dalam ERD dapat dijelaskan sebagai berikut: *SISWA* menunjukkan objek siswa, yaitu orang yang belajar atau kursus di masing-masing cabang PRIBADI (lihat e*ntity PRIBADI_CABANG*) dengan kelas (lihat *entity KELAS*) tertentu. *Entity* ini mengandung atribut id_siswa, nama_siswa, alamat_siswa, tanggal_lahir, umur_siswa, jenis_kelamin_siswa, dan no_telp_siswa.

a. *Entity KELAS*: menunjukkan objek kelas, yaitu tingkatan atau program yang diambil oleh siswa (lihat *entity SISWA*), seperti *basic, intermediate, advanced*, dan lain-lain. *Entity* ini mengandung atribut id_kelas, nama_kelas, kapasitas, waktu, jumlah_siswa, dan biaya.
b. *Entity PENGAJAR*: menunjukkan objek pengajar, yaitu orang yang mengajar di masing-masing cabang PRIBADI (lihat *entity PRIBADI_CABANG*) dengan kelas (lihat *entity KELAS*) tertentu. *Entity* ini mengandung atribut id_pengajar, nama_pengajar, alamat_pengajar, dan no_telp_pengajar.
c. *Entity PEMBAYARAN*: menunjukkan objek pembayaran, yaitu *entity* yang berfungsi untuk mengatur masalah pembayaran dari siswa. *Entity* ini mengandung atribut jenis_pembayaran, tanggal_pembayaran, angsuran_ke, term, dan status.
d. *Entity PRIBADI_CABANG*: menunjukkan objek kantor cabang PRIBADI, yaitu tempat kursus sekaligus tempat untuk mengatur masalah administrasi siswa. *Entity* ini mengandung atribut id_cabang, nama_cabang, alamat_cabang, dan no_telp_cabang.
e. *Entity ADMINISTRATOR*: menunjukkan objek administrator, yaitu orang yang mengelola sistem informasi di masing-masing cabang PRIBADI. Entity ini mengandung atribut id_admin, nama_admin, username, dan password.
f. *Relationship Belajar di Kelas*: menunjukkan hubungan antara objek kelas dan siswa. Relasi ini mengandung arti bahwa sebuah kelas dapat menerima lebih dari seorang siswa (*one to many*).
g. *Relationship Mengajar di Kelas*: menunjukkan hubungan antara objek



Gambar 5. Entity Relationship Diagram (ERD)

kelas dan pengajar. Relasi ini mengandung arti bahwa seorang pengajar dapat mengajar lebih dari satu kelas atau satu kelas dapat diajar oleh seorang pengajar (*many to one*).

h. *Relationship Membayar*: menunjukkan hubungan antara objek siswa dan pembayaran. Relasi ini mengandung arti bahwa seorang siswa dapat membayar dengan lebih dari satu jenis pembayaran (misalnya di term pertama siswa membayar dengan tunai sedangkan di term berikutnya membayar secara kredit) atau sebaliknya satu jenis pembayaran dapat dilakukan oleh lebih dari satu siswa (*many to many*).

i. *Relationship Belajar di Cabang*: menunjukkan hubungan antara objek kantor cabang PRIBADI dan siswa. Relasi ini mengandung arti bahwa satu cabang dapat menampung lebih dari satu orang siswa (*one to many*).

j. *Relationship Terdapat di* : menunjukkan hubungan antara objek kelas dan kantor cabang PRIBADI. Relasi ini mengandung arti bahwa satu kelas (dalam hal ini tingkatan, lihat keterangan *entity KELAS*) terdapat pada lebih dari satu cabang atau pada satu cabang terdapat lebih dari satu kelas (*many to many*).

k. *Relationship Mengajar di Cabang*: menunjukkan hubungan antara objek pengajar dan kantor cabang PRIBADI. Relasi ini mengandung arti bahwa seorang pengajar dapat mengajar di lebih dari satu cabang atau pada satu cabang diajar oleh lebih dari satu orang pengajar (*many to many*).

l. *Relationship Membayar di*: menunjukkan hubungan antara objek kantor cabang PRIBADI dan pembayaran. Relasi ini mengandung arti bahwa pembayaran untuk setiap siswa hanya dapat dilakukan di satu cabang, di mana siswa tersebut belajar.

m. *Relationship Mengelola*: menunjukkan hubungan antara objek kantor cabang PRIBADI dan administrator. Relasi ini mengandung arti bahwa di satu cabang bisa terdapat lebih dari satu administrator.

Perancangan Basis Data Secara Logika

Berdasarkan ERD Sistem Informasi Kesiswaan, dilakukan pemetaan dan normalisasi sehingga didapat hasil pemetaan *Entity Relationship Diagram* (ERD) seperti yang terlihat pada Gambar 6 sebagai berikut :

A. Pemetaan

SISWA

id_siswa	nama_siswa	alamat_siswa	tanggal_lahir	umur_siswa	jenis_kelamin_siswa	no_telp_siswa	id_kelas	id_cabang

Keterangan:
- id_siswa : nomor siswa
- nama_siswa : nama siswa yang bersangkutan
- alamat_siswa : alamat siswa
- tanggal_lahir : tanggal lahir siswa
- umur_siswa : umur siswa, bisa didapatkan dari atribut tanggal lahir siswa (*derived attribute*)
- jenis_kelamin : jenis kelamin siswa
- no_telp_siswa : nomor telepon siswa untuk mempermudah komunikasi

PEMBAYARAN

id_pembayaran	jenis_pembayaran	tgl_pembayaran	angsuran_ke	term	status	id_cabang

Keterangan :
- id_pembayaran : nomor pembayaran
- jenis_pembayaran : jenis pembayaran, tunai atau kredit
- tgl_pembayaran : tanggal pembayaran berlangsung
- angsuran_ke : angsuran ke berapa jika pembayaran dilakukan dengan kredit
- term : term ke berapa, semacam masa semester belajar
- status : status pembayaran, lunas atau belum

KELAS

id_kelas	nama_kelas	kapasitas	waktu	jumlah_siswa	biaya	id_pengajar

Keterangan :
- id_kelas : id kelas, mencakup nomor kelas juga
- nama_kelas : nama kelas tersebut, kelas *kindergarten,* TOEFL dll.
- kapasitas : kapasitas kelas tersebut
- waktu : waktu belajar di kelas tersebut
- jumlah_siswa : jumlah siswa yang belajar di kelas tersebut
- biaya : biaya belajar di kelas tersebut, misalnya kelas TOEFL biayanya berapa

PRIBADI_CABANG

id_cabang	nama_cabang	alamat_cabang	no_telp_cabang

Keterangan :
- id_cabang : id suatu kantor cabang
- nama_cabang : nama kantor cabang, misalnya Margonda dll.
- alamat_cabang : alamat cabang tersebut
- no_telp_cabang : nomor telepon cabang untuk mempermudah komunkasi

PENGAJAR

id_pengajar	nama_pengajar	alamat_pengajar	no_telp_pengajar	id_admin

Keterangan :
- id_pengajar : id pengajar
- nama_pengajar : nama pengajar yang bersangkutan
- alamat_pengajar : alamat tinggal pengajar
- no_telp_pengajar : nomor telepon pengajar untuk mempermudah komunikasi

ADMINISTRATOR

id_admin	nama_admin	*username*	*password*	id_cabang

Keterangan :
- id_admin : id administrator
- nama_admin : nama administrator tersebut
- *username* : *username* administrator
- *password* : *password* administrator

MEMBAYAR

id_siswa	id_kelas	id_cabang	id_pembayaran

TERDAPAT DI

id_kelas	Id_cabang

MENGAJAR DI CABANG

id_cabang	Id_pengajar

B. Normalisasi

SISWA

id_siswa	Nama_siswa	alamat_siswa	tanggal_lahir	umur_siswa	jenis_kelamin_siswa	no_telp_siswa	id_kelas	id_cabang

PEMBAYARAN

id_pembayaran	jenis_pembayaran	tgl_pembayaran	angsuran_ke	term	status	id_cabang

KELAS

id_kelas	no_kelas	nama_kelas	kapasitas	waktu	jumlah_siswa	biaya	id_pengajar

PRIBADI _CABANG

id_cabang	nama_cabang	alamat_cabang	No_telp_cabang

PENGAJAR

id_pengajar	nama_pengajar	alamat_pengajar	no_telp_pengajar

ADMINISTRATOR

id_admin	nama_admin	username	Password	id_cabang

MEMBAYAR

Id_siswa	id_kelas	id_cabang	id_pembayaran	no_kelas

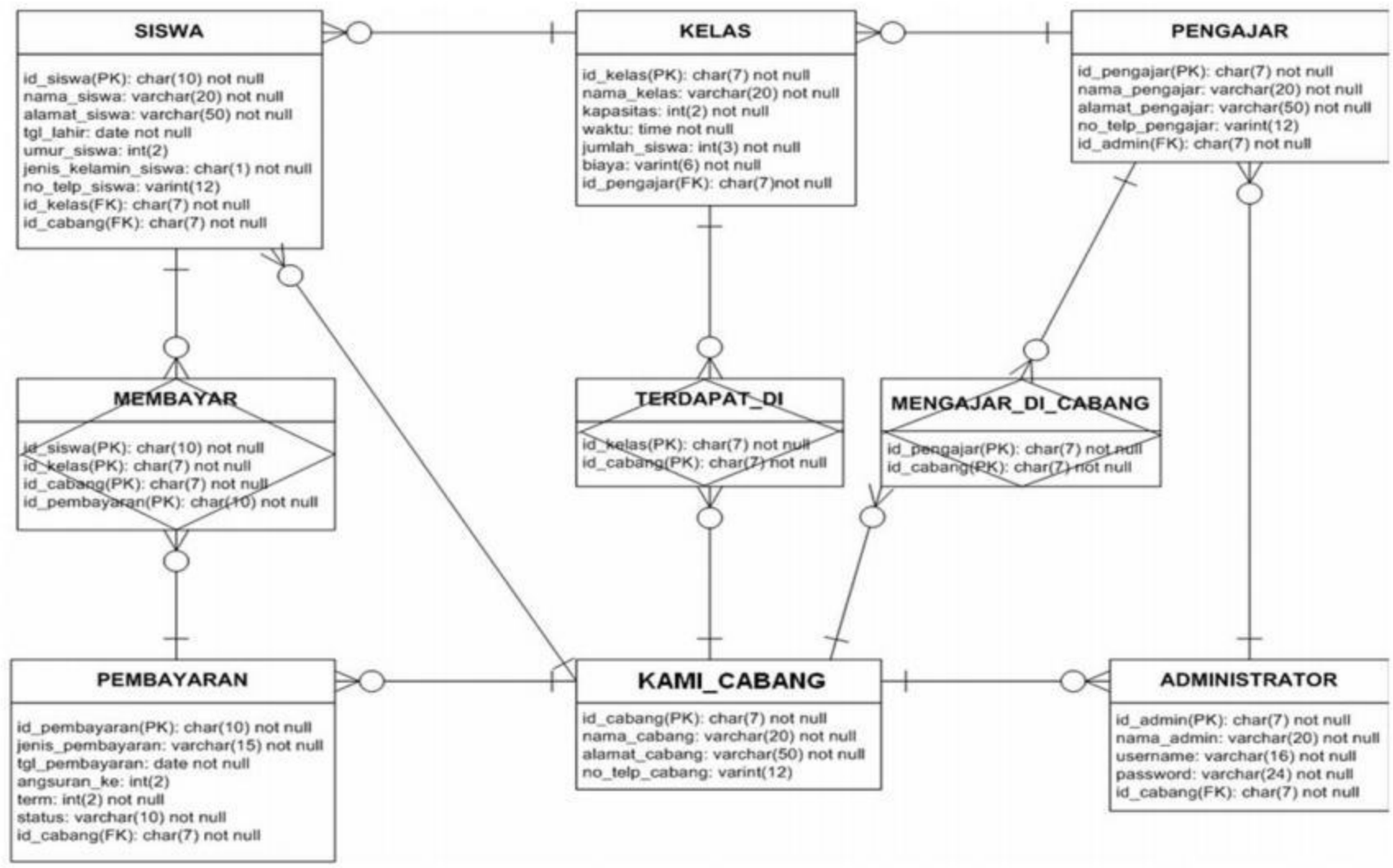
TERDAPAT_DI

id_kelas	id_cabang	no_kelas

MENGAJAR_DI_CABANG

id_cabang	Id_pengajar

Perancangan Database Secara Fisik



Gambar 6. Physical ERD

SIMPULAN

Dibangunnya basis data untuk proses penerimaan siswa, daftar ulang, pembayaran, dan pindah cabang pada lembaga ini, akan memudahkan dalam pencarian, pemakaian, serta penyusunan datanya. Berdasarkan perancangan yang telah dilakukan maka dihasilkan suatu basis data kesiswaan yang terdiri atas 9 tabel yaitu tabel cabang, administrator, pembayaran, pengajar, kelas, siswa, membayar, terdapat_di, dan mengajar_di_cabang.

DAFTAR PUSTAKA

Date, C.J, 1990. *An Introduction to Database System*, Addison Wesley Publishing Company, Vol. 1 & Vol. 2, New York.

Hartono, Jogiyanto, 1999. *Analisis dan Desain Sistem Informasi*,Yogyakarta: Andi Offset.

Whitten, L. Jeffrey, Bentley, D. Lonnie, Dittman, C. Kevin. 2004. *Systems Analysis and Design Methods*. New York: *McGraw-Hill Companie,*

Mc Fadden, F; Hoffer, Jeffrey A. 1991. *Database Management,* Third Edition. California: Benjamin/ Publishing Company Inc..

Connoly, Thomas; Begg, Carolyn; Strachan, Anne. 1996, *Database System: A Practical Approach to Design, Implementation and Management*. Addison Wesley,

Implementasi Sistem Basis Data (DBMS ORACLE)

Data Definition Language

```
SQL> create table cabang(
  2  id_cabang varchar2(7) not null,
  3  nama_cabang varchar2(20) not null,
  4  alamat_cabang varchar2(50) not null,
  5  no_telp_cabang varchar2(12),
  6  primary key (id_cabang));

Table created.
```

```
SQL> create table administrator(
  2  id_admin varchar2(7) not null,
  3  nama_admin varchar2(20) not null,
  4  username varchar2(16) not null,
  5  password varchar2(24) not null,
  6  id_cabang varchar2(7) not null,
  7  primary key (id_admin),
  8* foreign key (id_cabang) references cabang(id_cabang))

Table created.
```

```
SQL> create table pembayaran(
  2  id_pembayaran varchar2(10) not null,
  3  jenis_pembayaran varchar2(15) not null,
  4  tgl_pembayaran date not null,
  5  angsuran_ke int,
  6  term int,
  7  status varchar2(10) not null,
  8  id_cabang varchar2(7) not null,
  9  primary key (id_pembayaran),
 10* foreign key (id_cabang) references cabang(id_cabang))

Table created.
```

```
SQL> create table pengajar(
  2  id_pengajar varchar2(7) not null,
  3  nama_pengajar varchar2(20) not null,
  4  alamat_pengajar varchar2(50) not null,
  5  no_telp_pengajar varchar2(12),
  6  id_admin varchar2(7) not null,
  7  primary key (id_pengajar),
  8* foreign key (id_admin) references administrator(id_admin))

Table created.
```

```
SQL> create table kelas(
  2  id_kelas varchar2(7) not null,
  3  nama_kelas varchar2(20) not null,
  4  kapasitas int not null,
  5  jumlah_siswa int not null,
  6  biaya int not null,
  7  id_pengajar varchar2(7) not null,
  8  primary key (id_kelas),
  9* foreign key (id_pengajar) references pengajar(id_pengajar))

Table created.
```

```
SQL> create table siswa(
  2  id_siswa varchar2(10) not null,
  3  nama_siswa varchar2(20) not null,
  4  alamat_siswa varchar2(50) not null,
  5  tgl_lahir date not null,
  6  umur_siswa int,
  7  jenis_kelamin_siswa varchar2(1) not null,
  8  no_telp_siswa varchar2(12),
  9  id_kelas varchar2(7) not null,
 10  id_cabang varchar2(7) not null,
 11  primary key (id_siswa),
 12  foreign key (id_kelas) references kelas(id_kelas),
 13* foreign key (id_cabang) references cabang(id_cabang))

Table created.
```

```
SQL> create table membayar(
  2  id_siswa varchar2(10) not null,
  3  id_kelas varchar2(7) not null,
  4  id_cabang varchar2(7) not null,
  5  id_pembayaran varchar2(10) not null,
  6  foreign key (id_siswa) references siswa(id_siswa),
  7  foreign key (id_kelas) references kelas(id_kelas),
  8  foreign key (id_cabang) references cabang(id_cabang),
  9  foreign key (id_pembayaran) references pembayaran(id_pembayaran))

Table created.
```

```
SQL> create table terdapat_di(
  2  id_kelas varchar2(7) not null,
  3  id_cabang varchar2(7) not null,
  4  foreign key (id_kelas) references kelas(id_kelas),
  5  foreign key (id_cabang) references cabang(id_cabang));

Table created.
```

```
SQL> create table mengajar_di_cabang(
  2  id_pengajar varchar2(7) not null,
  3  id_cabang varchar2(7) not null,
  4  foreign key (id_pengajar) references pengajar(id_pengajar),
  5  foreign key (id_cabang) references cabang(id_cabang));

Table created.
```

```
SQL> desc siswa;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_SISWA                                  NOT NULL VARCHAR2(10)
NAMA_SISWA                                NOT NULL VARCHAR2(20)
ALAMAT_SISWA                              NOT NULL VARCHAR2(50)
TGL_LAHIR                                 NOT NULL DATE
UMUR_SISWA                                         NUMBER(38)
JENIS_KELAMIN_SISWA                       NOT NULL VARCHAR2(1)
NO_TELP_SISWA                                      VARCHAR2(12)
ID_KELAS                                  NOT NULL VARCHAR2(7)
ID_CABANG                                 NOT NULL VARCHAR2(7)
```

```
SQL> desc kelas;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_KELAS                                  NOT NULL VARCHAR2(7)
NAMA_KELAS                                NOT NULL VARCHAR2(20)
KAPASITAS                                 NOT NULL NUMBER(38)
JUMLAH_SISWA                              NOT NULL NUMBER(38)
BIAYA                                     NOT NULL NUMBER(38)
ID_PENGAJAR                               NOT NULL VARCHAR2(7)
```

```
SQL> desc pengajar;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_PENGAJAR                               NOT NULL VARCHAR2(7)
NAMA_PENGAJAR                             NOT NULL VARCHAR2(20)
ALAMAT_PENGAJAR                           NOT NULL VARCHAR2(50)
NO_TELP_PENGAJAR                                   VARCHAR2(12)
ID_ADMIN                                  NOT NULL VARCHAR2(7)
```

```
SQL> desc administrator;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_ADMIN                                  NOT NULL VARCHAR2(7)
NAMA_ADMIN                                NOT NULL VARCHAR2(20)
USERNAME                                  NOT NULL VARCHAR2(16)
PASSWORD                                  NOT NULL VARCHAR2(24)
ID_CABANG                                 NOT NULL VARCHAR2(7)
```

```
SQL> desc cabang;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_CABANG                                 NOT NULL VARCHAR2(7)
NAMA_CABANG                               NOT NULL VARCHAR2(20)
ALAMAT_CABANG                             NOT NULL VARCHAR2(50)
NO_TELP_CABANG                                     VARCHAR2(12)
```

```
SQL> desc pembayaran;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_PEMBAYARAN                             NOT NULL VARCHAR2(10)
JENIS_PEMBAYARAN                          NOT NULL VARCHAR2(15)
TGL_PEMBAYARAN                            NOT NULL DATE
ANGSURAN_KE                                        NUMBER(38)
TERM                                               NUMBER(38)
STATUS                                    NOT NULL VARCHAR2(10)
ID_CABANG                                 NOT NULL VARCHAR2(7)
```

```
SQL> desc membayar;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_SISWA                                  NOT NULL VARCHAR2(10)
ID_KELAS                                  NOT NULL VARCHAR2(7)
ID_CABANG                                 NOT NULL VARCHAR2(7)
ID_PEMBAYARAN                             NOT NULL VARCHAR2(10)
```

```
SQL> desc terdapat_di;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_KELAS                                  NOT NULL VARCHAR2(7)
ID_CABANG                                 NOT NULL VARCHAR2(7)
```

```
SQL> desc mengajar_di_cabang;
Name                                      Null?    Type
----------------------------------------- -------- ----------------
ID_PENGAJAR                               NOT NULL VARCHAR2(7)
ID_CABANG                                 NOT NULL VARCHAR2(7)
```