PERANCANGAN DATABASE PENDAFTARAN SWAB ANTIGEN BERBASIS ONLINE

DESIGN OF ONLINE-BASED ANTIGEN SWAB REGISTRATION DATABASE

Erfanti Fatkhiyah[1*],Joko Triyono[2], Eko Nur Cahyo[3]

[1,3]Jurusan Informatika, Fakultas Teknologi Industri, Institut Sains dan Teknologi AKPRIND Yogyakarta
[2]Jurusan Rekayasa Sistem Komputer, Fakultas Sains Terapan, Institut Sains dan Teknologi AKPRIND Yogyakarta
[1*]erfanti@akprind.ac.id, [2]jack@akprind.ac.id, [3]ekonurcahyop29@gmail.com
***penulis korespondensi**

Abstrak

Swab adalah cara untuk memperoleh bahan pemeriksaan (sampel). Swab dilakukan pada nasofaring dan atau orofarings. Hampir semua kegiatan masyarakat mensyaratkan bahwa harus membawa bukti Swab Antigen dengan hasil Negatif, sehingga kebutuhan akan perlengkapan swab tersebut sangat melonjak tinggi dan klinik. Karena pentingnya *Swab* tersebut, maka secara bisnis sangat menggiurkan. Bahkan beberapa waktu yang lalu telah viral tentang penyalahgunaan swab tersebut dengan melakukan hal-hal yang tidak memenuhi SOP bahkan melanggar, bahkan tidak tanggung-tanggung kelompok pelaku tersebut bisa meraup sampai puluhan juta. Namun dari sekian banyak sistem online yang ada, masih lebih banyak mengedepankan bagaimana proses mendaftar dan kebanyakan tanpa menginformasikan berapa stok atau kapasitas bahan dan alat yang tersedia secara transparan. Penelitian ini bertujuan untuk melakukan pemodelan dalam melakukan perancangan arsitektur database pendaftaran swab antigen berbasis online untuk membantu dalam pengembangan aplikasi pendaftaran swab antigen.

Kata Kunci: Arsitektur, Database, Permodelan, Swab

Abstract

Swab is a way to obtain examination material (sample). The swab is performed on the nasopharynx and/or oropharynx. Almost all community activities require that they must bring evidence of an antigen swab with a negative result, so the need for swab equipment is very high and in clinics. Because of the importance of the Swab, it is very lucrative in business. Even some time ago it was viral about the misuse of the swab by doing things that did not meet the SOP and even violated it, even though this group of perpetrators could earn up to tens of millions. However, of the many existing online systems, there are still more that put forward the process of registering and mostly without informing how much stock or capacity of materials and tools are available transparently. This study aims to conduct modeling in designing an online-based antigen swab registration database architecture to assist in the development of antigen swab registration applications

Keywords: Architecture, Database, Modeling, Swab

1. PENDAHULUAN

Swab adalah cara untuk memperoleh bahan pemeriksaan (sampel). Swab dilakukan pada nasofaring dan atau orofarings. Pengambilan ini dilakukan dengan cara mengusap rongga nasofarings dan atau orofarings dengan menggunakan alat seperti kapas lidi khusus. Swab antigen adalah tes diagnostik cepat Covid-19 yang dilakukan untuk mendeteksi keberadaan antigen virus

corona pada sampel yang berasal dari saluran pernapasan. Antigen ini akan terdeteksi ketika virus aktif bereplikasi.

Hampir semua kegiatan masyarakat mensyaratkan bahwa harus membawa bukti Swab Antigen dengan hasil Negatif, sehingga kebutuhan akan perlengkapan swab tersebut sangat melonjak tinggi dan klinik atau pelayanan swab juga dibutuhkan sangat-sangat banyak.

Karena pentingnya *Swab* tersebut, maka secara bisnis sangat menggiurkan. Bahkan beberapa waktu yang lalu telah viral tentang penyalahgunaan swab tersebut dengan melakukan hal-hal yang tidak memenuhi SOP bahkan melanggar, bahkan tidak tanggung-tanggung kelompok pelaku tersebut bisa meraup sampai puluhan juta.

Proses Swab juga harus tetap menjaga protocol kesehatan, sehingga kerumunan di tempat swab harus dihindari, sehingga banyak pelayanan swab yang membuka layanan melalui sistem online. Namun dari sekian banyak sistem online yang ada, masih lebih banyak mengedepankan bagaimana proses mendaftar dan kebanyakan tanpa menginformasikan berapa stok atau kapasitas bahan dan alat yang tersedia secara transparan. Realitanya seseorang bisa melakukan test Swab seperti ini berkali-kali dan biasanya akan melakukan test swab ditempat yang telah biasa dilakukan jika memungkinkan. Sehingga record swab yang dilakukan serta hasilnya harusnya tersimpan secara sistemik dan bisa diakses oleh konsumen secara mandiri dan bersifat pribadi.

Penelitian ini bertujuan untuk melakukan pemodelan dalam melakukan perancangan arsitektur database pendaftaran swab antigen berbasis online untuk membantu dalam pengembangan aplikasi pendaftaran swab antigen yang merupakan bagian dari penelitian pemodelan dan perancangan arsitektur aplikasi untuk menginformasikan kepada khalayak stok atau kapasistas Swab antigen pada halaman web, melakukan pendaftaran secara online bagi konsumen baru atau member serta memberikan akses pribadi kepada member untuk melihat histori test yang telah dilakukan. Proses pendaftaran dengan memilih jam kedatangan serta bisa melihat posisi antrian secara realtime sehingga akan mengurangi terjadinya kerumunan dalam antrian serta hasilnya akan langsung dikirimkan ke sistem informasi dan kontak member.

2. DASAR TEORI / MATERIAL DAN METODOLOGI/PERANCANGAN

2.1 API (*Application Programming Interface)*

Dalam sebuah penelitian di peroleh hasil tentang pengembangan sebuah *prototype* sistem informasi yang di kombinasikan dengan jejaring sosial *twitter*, dimana jejaring *twitter* digunakan petani untuk melaporkan semua kegiatannya ke sistem informasi, dengan menggunakan fasilitas *APIs* (*Aplication Programming Language*) maka informasi yang masuk akan di kirimkan ke sistem informasi dengan menggunakan *account* dari *twitter* pengirim. Dengan metode ini secara teknologi dan biaya petani tidak mengalami kesulitan dalam melaporkan kegiatannya, sedangkan dari sisi investor akan bisa melihat perkembangan investasinya. [1]

2.2. PHP: Hypertext Prepocessor (PHP)

PHP adalah Bahasa scripting yang menyatu dengan HTML (kode dasar website) dan dijalankan pada *server side.* Artinya, semua sintaks PHP yang diberikan akan sepenuhnya dijalankan pada server, sedangkan yang dikirimkan ke browser hanya hasilnya saja. Menurut (Zal, 2018) Konsep kerja PHP diawali dengan permintaan suatu halaman web (file.php) oleh browser atau klien. Kemudian berdasarkan alamat di Internet (URL), browser mendapatkan alamat dari web server, yang akan mengidentifikasi halaman yang diminta, ketika file PHP yang diminta didapatkan oleh web server, isinya segera dikirimkan ke mesin PHP untuk diproses dan memberikan hasilnya (berupa kode HTML) ke web server, lalu menyampaikannya ke klien. [2]

2.3 *Unified Modeling Language* (UML)

UML singkatan dari *Unified Modeling Language* (UML) yang berarti bahasa pemodelan standart. Ketika membuat model menggunakan konsep *Unified Modeling Language* (UML) ada aturan-aturan yang harus diikuti. Elemen pada model-model yang dibuat berhubungan satu dengan lainnya harus mengikuti standar yang ada. *Unified Modeling Language* (UML) bukan hanya sekadar diagram, tetapi juga menceritakan konteksnya [3].

2.4 My Structured Query Language (MYSQL)

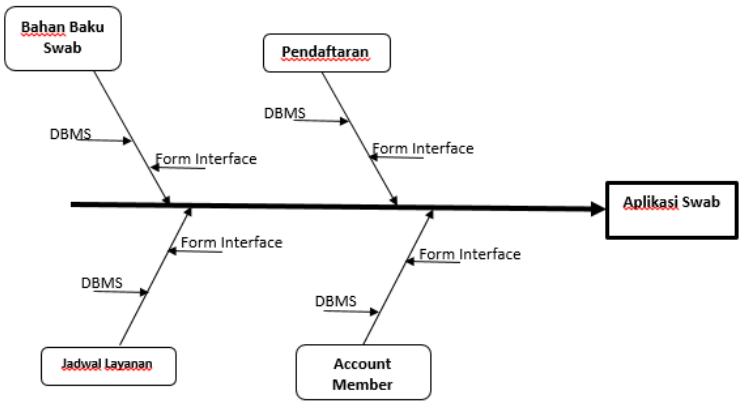
MySQL merupakan program aplikasi untuk membuat suatu DBMS (*DataBase Management System*) yang berbasis SQL (*Structured Query Language*). MySQL mempergunakan lisensi GPL (*GNU General Public License*). Pada sebuah basis data yang dibuat oleh MySQL mengandung satu atau beberapa tabel. Tabel terdiri atas sejumlah baris dan kolom. MySQL mempunyai beberapa kelebihan dibandingkan dengan yang lainnya, seperti PostgresSQL, MicrosoftSQL Server, dan Oracle. Kelebihan MySQL adalah pada kecepatan akses, biaya, konfigurasi dan tersedia *source code* karena MySQL berada di bawah *Open Source License.* MySQL mempunyai perintah-perintah yang tergolong DML (*Data Manipulation Language*). DML adalah suatu bahasa yang digunakan untuk memanipulasi data, seperti menambah, menghapus, menampilkan, dan mengubah suatu data. Perintah-perintah itu adalah *insert, update, delete dan select.* [2]

2.5 Metodologi

Metode penelitian dilakukan dalam skala laboratorium di Laboratorium Jaringan dan Internet Institut Sains dan Teknologi AKPRIND Yogyakarta, data atau model yang digunakan dalam penelitian ini bersifat imaginer yang disimulasikan di dalam skala laboratorium dan juga diimplementasikan secara online.

2.5.1 Rancangan Sistem

Sistem Informasi yang akan dibangun dalam hal ini adalah sistem imaginer kegiatan publikasi ketersediaan bahan baku Swab antigen, Jadwal pelayanan Swab, Pendaftaran Swab Online, pengiriman data hasil swab tersistem dan menggunakan media kontak pelanggan serta pelanggan memiliki account pada sistem informasi untuk melihat histori layanan yang sudah dilakukan digambarkan pada fishone diagram seperti terlihat pada gambar 1



Gambar 1. Fisbhone Diagram

2.5.2 Identifikasi Kebutuhan Modul

Modul yang perlu dipersiapkan sesuai dengan gambar 1 yaitu

Database adalah *database* yang digunakan untuk menampung transaksi dari kegiatan sistem informasi.

Bahan Baku Swab Antigen, adalah ketersediaan bahan baku untuk melayani swab, system ini akan mencatat semua transaksi pengadaan bahan baku swab.

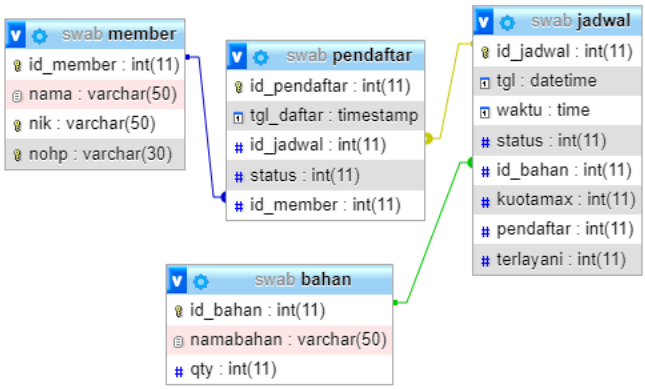
Jadwal Layanan, adalah jadwal layanan yang dibuka untuk melayani pelanggan.

Pendaftaran Swab Online adalah sistem pendaftaran swab melalui media sistem informasi secara online sehingga pelanggan akan datang sesuai dengan jadwal yang telah ditentukan.

Hasil Swab adalah informasi hasil test swab secara sistemik melalui sistem informasi dan juga melalui kontak pelanggan.

2.5.3 Design Database

Untuk implementasi rancangan tersebut diperlukan sebuah database seperti pada gambar 2.



Gambar 2. Design Database

Pada design ini disertakan pula trigger untuk merangkum jumlah pendaftar pada tabel jadwal, dan juga penerapan constraint check untuk mengatur agar pendaftar tidak melampaui kuotamax.

3. PEMBAHASAN

Dalam pengujian ini dilakukan simulasi skala laboratorium baik dalam merancang database maupun merancang aplikasi transaksional.

3.1 Database

Rancangan database seperti gambar 2, dengan perincian struktur tabel, index, relasi sebagai berikut.

3.1.1 Struktur dari tabel bahan

```
CREATE TABLE `bahan` (
  `id_bahan` int(11) NOT NULL,
  `namabahan` varchar(50) NOT NULL,
  `qty` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3.1.2 Struktur dari tabel `jadwal`

```
CREATE TABLE `jadwal` (
  `id_jadwal` int(11) NOT NULL,
  `tgl` datetime NOT NULL,
  `waktu` time NOT NULL,
  `status` int(11) NOT NULL DEFAULT 0,
  `id_bahan` int(11) NOT NULL,
```

```
  `kuotamax` int(11) NOT NULL DEFAULT 0,
  `pendaftar` int(11) NOT NULL DEFAULT 0,
  `terlayani` int(11) NOT NULL DEFAULT 0
) ;
```

3.1.2 Struktur dari tabel `member`

```
CREATE TABLE `member` (
  `id_member` int(11) NOT NULL,
  `nama` varchar(50) NOT NULL,
  `nik` varchar(50) NOT NULL,
  `nohp` varchar(30) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3.1.3 Struktur dari tabel `pendaftar`

```
CREATE TABLE `pendaftar` (
  `id_pendaftar` int(11) NOT NULL,
  `tgl_daftar` timestamp NOT NULL DEFAULT current_timestamp(),
  `id_jadwal` int(11) NOT NULL,
  `status` int(11) NOT NULL,
  `id_member` int(11) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

3.1.4 Indeks untuk tabel

```
ALTER TABLE `bahan`
  ADD PRIMARY KEY (`id_bahan`);
ALTER TABLE `jadwal`
  ADD PRIMARY KEY (`id_jadwal`),
  ADD KEY `id_bahan` (`id_bahan`);
ALTER TABLE `member`
  ADD PRIMARY KEY (`id_member`),
  ADD UNIQUE KEY `nik` (`nik`),
  ADD UNIQUE KEY `nohp` (`nohp`);
ALTER TABLE `pendaftar`
  ADD PRIMARY KEY (`id_pendaftar`),
  ADD KEY `id_member` (`id_member`),
  ADD KEY `id_jadwal` (`id_jadwal`);
```

3.1.5 AUTO_INCREMENT untuk tabel

```
ALTER TABLE `bahan`
  MODIFY `id_bahan` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE `jadwal`
  MODIFY `id_jadwal` int(11) NOT NULL AUTO_INCREMENT;

ALTER TABLE `member`
  MODIFY `id_member` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=2;

ALTER TABLE `pendaftar`
  MODIFY `id_pendaftar` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=5;
```

3.1.6 RELASI TABEL

```
ALTER TABLE `jadwal`
  ADD CONSTRAINT `jadwal_ibfk_1` FOREIGN KEY (`id_bahan`) REFERENCES
`bahan` (`id_bahan`) ON DELETE NO ACTION ON UPDATE NO ACTION;

ALTER TABLE `pendaftar`
  ADD CONSTRAINT `pendaftar_ibfk_1` FOREIGN KEY (`id_member`) REFERENCES
`member` (`id_member`) ON DELETE NO ACTION ON UPDATE NO ACTION,
```

```
  ADD CONSTRAINT `pendaftar_ibfk_2` FOREIGN KEY (`id_jadwal`) REFERENCES
`jadwal` (`id_jadwal`) ON DELETE NO ACTION ON UPDATE NO ACTION;
```

3.2 *Trigger*

Trigger digunakan untuk membantu keamanan transaksi, dan memudahkan pengembang aplikasi dalam melakukan validasi. Dalam perancangan ini digunakan 2 buah *trigger*.

***Trigger* AddPendaftar**

Trigger ini untuk merekam tiap ada pendaftar baru pada sebuah jadwal, sehingga tiap ada pendaftar baru maka jadwal pendaftar akan bertambah satu.

```
DELIMITER $$
CREATE TRIGGER `addpendaftar` AFTER INSERT ON `pendaftar` FOR EACH ROW
update jadwal set pendaftar=pendaftar+1 WHERE id_jadwal=new.id_jadwal
$$
DELIMITER ;
```

Pengujian *trigger*

Kondisi awal sebelum dilakukan pengujian pada tabel jadwal dan pendaftar seperti pada gambar 3, setelah dilakukan penambahan data 1 pendaftar pada tabel pendaftar, maka kondisi tabel jadwal dan pendaftar seperti terlihat pada gambar 4.

Terlihat di tabel jadwal secara otomatis sudah akan bertambah 1 pada field pendaftar.

```
MariaDB [swab]> select * from jadwal;
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
| id_jadwal | tgl                 | waktu    | status | id_bahan | kuotamax | pendaftar | terlayani |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
|         1 | 2021-12-10 08:00:00 | 00:59:00 |      0 |        1 |        5 |         1 |         0 |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
1 row in set (0.000 sec)

MariaDB [swab]> select * from pendaftar;
+--------------+---------------------+-----------+--------+-----------+
| id_pendaftar | tgl_daftar          | id_jadwal | status | id_member |
+--------------+---------------------+-----------+--------+-----------+
|            1 | 2021-12-09 20:54:50 |         1 |      0 |         1 |
+--------------+---------------------+-----------+--------+-----------+
1 row in set (0.000 sec)
```

Gambar 3 Kondisi awal

```
MariaDB [swab]> insert into pendaftar(id_jadwal,id_member) values(1,2);
Query OK, 1 row affected, 1 warning (0.223 sec)

MariaDB [swab]> select * from pendaftar;
+--------------+---------------------+-----------+--------+-----------+
| id_pendaftar | tgl_daftar          | id_jadwal | status | id_member |
+--------------+---------------------+-----------+--------+-----------+
|            1 | 2021-12-09 20:54:50 |         1 |      0 |         1 |
|            5 | 2021-12-09 21:48:52 |         1 |      0 |         2 |
+--------------+---------------------+-----------+--------+-----------+
2 rows in set (0.000 sec)

MariaDB [swab]> select * from jadwal;
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
| id_jadwal | tgl                 | waktu    | status | id_bahan | kuotamax | pendaftar | terlayani |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
|         1 | 2021-12-10 08:00:00 | 00:59:00 |      0 |        1 |        5 |         2 |         0 |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
```

Gambar 4 Kondisi setelah dilakukan penambahan data

Trigger delPendaftar

Trigger ini digunakan untuk mengurangi data pendaftar pada tabel jadwal jika ada pendaftar yang membatalkan pendaftaran.

```
DELIMITER $$
```

```
CREATE TRIGGER `delpendaftar` BEFORE DELETE ON `pendaftar` FOR EACH ROW
update jadwal set pendaftar=pendaftar-1 WHERE id_jadwal=old.id_jadwal
$$
DELIMITER ;
```

Pengujian *trigger*

Dengan kondisi akhir data seperti pada gambar 4, maka jika dilakukan pembatalan atau penghapusan data pada tabel pendaftar, maka jumlah data pada jadwal.pendaftar akan berkurang satu seperti terlihat pada gambar 5.

```
MariaDB [swab]> delete from pendaftar where id_pendaftar=1;
Query OK, 1 row affected (0.095 sec)

MariaDB [swab]> select * from pendaftar;
+--------------+---------------------+-----------+--------+-----------+
| id_pendaftar | tgl_daftar          | id_jadwal | status | id_member |
+--------------+---------------------+-----------+--------+-----------+
|            5 | 2021-12-09 21:48:52 |         1 |      0 |         2 |
+--------------+---------------------+-----------+--------+-----------+
1 row in set (0.001 sec)

MariaDB [swab]> select * from jadwal;
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
| id_jadwal | tgl                 | waktu    | status | id_bahan | kuotamax | pendaftar | terlayani |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
|         1 | 2021-12-10 08:00:00 | 00:59:00 |      0 |        1 |        5 |         1 |         0 |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
1 row in set (0.000 sec)
```

Gambar 5 Kondisi setelah dilakukan penghapusan data

3.3 *Constraint*

Constraint digunakan juga untuk membantu memvalidasi nilai dari data, dalam hal ini digunakan *constraint* untuk membatasi jumlah pendaftar tidak melebihi kuota yang disediakan.

Constraint

```
ALTER TABLE table_jadwal
ADD CONSTRAINT kuota CHECK ( kuotamax < pendaftar );
```

Pengujian *Constraint*

Pengujian *constraint* seperti terlihat pada gambar 6, dimana dengan kuota max 5 maka jika tetap akan ada penambahan pendaftar, secara sistem akan tertolak.

```
MariaDB [swab]> select * from pendaftar;
+--------------+---------------------+-----------+--------+-----------+
| id_pendaftar | tgl_daftar          | id_jadwal | status | id_member |
+--------------+---------------------+-----------+--------+-----------+
|            5 | 2021-12-09 21:48:52 |         1 |      0 |         2 |
|            6 | 2021-12-09 22:04:46 |         1 |      0 |         2 |
|            7 | 2021-12-09 22:04:50 |         1 |      0 |         3 |
|            8 | 2021-12-09 22:05:05 |         1 |      0 |         4 |
+--------------+---------------------+-----------+--------+-----------+
4 rows in set (0.000 sec)

MariaDB [swab]> select * from jadwal;
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
| id_jadwal | tgl                 | waktu    | status | id_bahan | kuotamax | pendaftar | terlayani |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
|         1 | 2021-12-10 08:00:00 | 00:59:00 |      0 |        1 |        5 |         4 |         0 |
+-----------+---------------------+----------+--------+----------+----------+-----------+-----------+
1 row in set (0.000 sec)

MariaDB [swab]> insert into pendaftar(id_jadwal,id_member) values(1,1);
ERROR 4025 (23000): CONSTRAINT `kuota` failed for `swab`.`jadwal`
MariaDB [swab]>
```

Gambar 6 Pengujian *Constraint*

4. KESIMPULAN

4.1 Kesimpulan

Dari hasil penelitian ini dapat disimpulkan bahwa proses perancangan database yang tepat dengan menambahkan *trigger, constraint* akan sangat membantu pengembang aplikasi dalam validasi data, sehingga tidak semua validasi dilakukan pada sisi aplikasi tetapi juga dilakukan pada sisi rdbms.

4.2 Rekomendasi

Adapun saran untuk pengembangan penelitian selanjutnya adalah penambahan view, procedure dan beberapa tool untuk meningkatkan kekuatan rancangan database.

DAFTAR PUSTAKA

[1] J. Triyono, "Sistem Informasi Agroteknologi berbasis Web dan Jejaring Sosial Twitter.," in *Seminar Nasional IENACO (pp. 205-212)*, Surakarta, 2015.

[2] Wardana, Aplikasi Website Profesional dengan PHP dan jQuery, Jakarta: PT Elex Media Komputindo, 2016.

[3] M. Muslihudin and Oktafianto, Analisis dan Perancangan Sistem Informasi Menggunakan Model Terstruktur dan UML, Yogyakarta: Penerbit Andi, 2016.

[4] E. Z. A. &. C. S. Winarno, Buku Sakti Pemrograman, Jakarta: PT. Elex Media Komputindo, 2013.