Implementasi Rancangan Database Akademik menggunakan Function, Store Procedure, Trigger dan View

Joko Triyono[1], Prafitra Dimas Akbar[2]
[1] *Program Studi Rekayasa Sistem Komputer, Fakultas Teknologi Informasi dan Bisnis, Institut Sains & Teknologi AKPRIND Yogyakarta*
[2] *Program Studi Informatika, Fakultas Teknologi Informasi dan Bisnis, Institut Sains & Teknologi AKPRIND Yogyakarta*
e-mail: [1]jack@akprind.ac.id, [2]prftrdms@gmail.com

Abstrak

Perkembangan sistem informasi selalu diimbangi dengan ketersediaan sistem database yang handal, sehingga suatu sistem informasi sangat membutuhkan rancangan RDBMS yang mumpuni. Dengan semakin kompleknya permasalahan yang dihadapi, maka dibutuhkan sebuah RDBMS yang bisa melayani berbagai sistem informasi dan bisa mengawal aturan bisnis, sehingga berbagai aturan bisnis dikelola oleh RDBMS sendiri dengan memanfaatkan function, store procedure, trigger dan view. Dalam penelitian ini telah diujicobakan pada sebuah rancangan database akademik untuk mengelola pengisian KRS dan pengisian NILAI dengan memfokuskan transaksional dilakukan pada ranah RDBMS, sehingga aplikasi hanya akan melakukan koneksi ke fasilitas-fasilitas tersebut tanpa koneksi langsung ke tabel. Dari hasil penelitian disimpulkan bahwa pengelolaan algoritma database yang tepat maka kesalahan dalam menginterprestasikan diagram relasionalship dapat diminimalisir, begitu juga dengan rule bisnis aplikasi akan lebih terjamin karena semua dikelola pada RDBMS. Dengan penerapan Store Procedure, Function, Trigger serta View yang tepat maka akan dihasilkan sebuah rancangan database yang kuat secara algoritma. Dengan metode ini, maka pada bagian pengembang aplikasi akan menjadi sangat mudah dan ringan karena tidak dibebani oleh diagram relasionalship maupun algoritma-algoritma dari rule bisnis aplikasi.

Kata kunci*—function, store procedure, trigger, view*

Abstrack (Bahasa Inggris)

The development of information systems is always balanced with the availability of a reliable database system, so an information system really needs a qualified RDBMS design. With the increasing complexity of the problems faced, an RDBMS is needed that can serve various information systems and can control business rules, so that various business rules are managed by the RDBMS itself by utilizing functions, store procedures, triggers and views. In this study, it has been tested on an academic database design to manage KRS filling and VALUE filling by focusing transactionally on the RDBMS realm, so that the application will only connect to these facilities without direct connection to the table. From the results of the study, it was concluded that the proper management of the database algorithm, errors in interpreting the relationalship diagram can be minimized, as well as the application business rules will be more guaranteed because all are managed in the RDBMS. With the proper implementation of Store Procedure, Function, Trigger and View, an algorithmically robust database design will be generated. With this method, the application developer will be very easy and lightweight because it is not burdened by relationalship diagrams or algorithms from application business rules.

Keywords*— function, store procedure, trigger, view*

1. PENDAHULUAN

Perkembangan sistem informasi sangat pesat, berbagai metode dalam pengembangan sistem informasi begitu banyaknya, begitu juga dengan perkembangan dalam metode perancangan database. Dalam database walau sudah banyak yang menggunakan NoSQL akan tetapi keberadaan RDBMS tetap masih sangat diperlukan. Kemampuan dalam menangani referensial integrity dan transaksional masih dominan di pegang oleh RDBMS. Pada sisi pengguna, kecepatan memperoleh informasi menjadi sangat penting, akan tetapi pengamanan transaksional menjadi hal yang tidak bisa di anggap sederhana. Beberapa algoritma dalam sebuah rule bisnis tentunya akan lebih baik jika bisa dikelola oleh RDBMS, sehingga bagian developer aplikasi tidak akan mengalami kesalahan terjemahan dalam algoritma. Untuk menjaga agar

algoritma transaksional tidak mengalami kesalahan terjemahan, maka pelu penerapan sebuah metode dimana proses transaksi baik POST maupun GET akan dilewatkan sebuah antarmuka di RDBMS dan tidak langsung bersinggungan dengan tabel-tabel pada RDBMS, akan tetapi akan dilewatkan Store Produre, Function, View maupun Trigger yang diatur sedemikian ruga sehingga developer tinggal menggunakan fasilitas tersebut untuk POST maupun GET.

Dengan semakin banyaknya table-tabel dan database dalam perkembangan sebuah sistem informasi, maka akan semakin komplek aturan relasionalship antar table, juga saat terjadi proses GET maupun POST akan melibatkan banyak tabel yang tentunya juga banyak aturan yang cukup komplek sehingga dibutuhkan solusi tepat yang akan menyederhanakan alur tersebut dan selalu terjamin ke absahan transaksionalnya. Dalam kasus ini, akan diambil sebuah perancangan RDBMS akademik, yang mana untuk transaksional pengisian Pengisian KRS maupun pengisian Hasil Ujian (nilai) dari sisi interface aplikasi tidak akan langsung melakukan transaksi ke tabel, akan tetapi akan melalui sebuah mekanisme tertentu melalui pemanggilan Store Procedure, Function dan View maupun Trigger.

Dalam sebuah buku dengan judul SQL The Complete Refference Third Edition [1] didefinisikan tentang konsep trigger yaitu semua kejadian apa pun yang menyebabkan perubahan dalam isi tabel, pengguna dapat menentukan tindakan terkait yang harus dilakukan oleh DBMS. Tiga peristiwa yang dapat memicu tindakan adalah upaya untuk menyisipkan, menghapus, atau memperbarui baris dari tabel. Tindakan yang dipicu oleh suatu peristiwa ditentukan oleh urutan pernyataan SQL Penelitian serupa oleh Joko Triyono [2]diperoleh kesimpulan bahwa dengan melakukan optimalisasi database akan memungkinkan para pengembang sistem informasi tidak terlalu direpotkan dengan algoritma SQL, karena banyak pekerjaan yang bisa di bebankan ke RDBMS, baik itu untuk transaksional maupun untuk view. Dengan metode ini, apapun bahasa pemrograman yang digunakan tetap akan mengakses SQL yang sangat sederhana, sehingga akan memudahkan dalam pengembangan sistem informasi dan tetap akan mendapatkan algoritma database yang sama. Pada penelitian yang lain Zanuar Alfiani [3] mengatakan perancangan sistem informasi merupakan hal yang sangat penting bagi organisasi, dengan rancangan sistem yang sesuai dengan kebutuhan dapat berpengaruh positif kepada kinerja organisasi, terlebih organisasi perusahaan dituntut untuk dapat melayani dengan cepat dan sigap. Juga pada penelitian yang lain Joko Triyono etc mengatakan bahwa perancangan distribusi data terdistribusi dan aplikasi menjadi tantangan [4] tersendiri karena informasi harus lebih mendekat kepada pemakai sehingga pemakai menjadi lebih fleksibel dalam mengelola informasi untuk menunjang kegiatan dan pekerjaan. Penggunaan replikasi database yang dikombinasikan dengan web service server menjadi salah satu alternatif agar pemakai kelompok transaksional dan manajerial bisa lebih leluasa dalam mengelola informasi. Pada penelitian lain tentang perancangan database Erfanti etc [5] menjelaskan bahwa 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. Joko Triyono dalam penelitian lain mengatakan bahwa [6] perancangan RDBMS harus bisa menangkap dan mengontrol semua kegiatan yang terjadi pada sebuah sistem informasi tanpa harus mengganggu bahkan menyentuh sisi aplikasi sistem informasi. Keamanan dan histori data sebuah sistem informasi harus selalu terjaga secara berkelanjutan dan tersistem, sehingga data tersebut bisa dianalisis untuk mendapatkan pola dalam pengembangan sistem selanjutnya. Penelitian ini menghasilkan pemodelan kontrol dalam sebuah sistem informasi transaksional berbasis RDBMS dengan menggunakan trigger dan waktu server untuk merekam semua kejadian dalam sebuah RDBMS sehingga diperoleh sebuah data perkembangan RDBMS dari waktu ke waktu yang bisa digunakan untuk analisis selanjutnya.

Dengan penelitian ini diharapkan bisa diperoleh metode yang efektif dalam pengembangan sebuah system informasi, dengan meletakkan algoritma database di sisi RDBMS, sehingga pihak pengembang aplikasi akan dimudahkan dalam melakukan POST maupun GET

informasi RDBMS dengan software apapun dan juga tidak akan terjadi salah arti dalam transaksional.

2. METODE PENELITIAN

Metode penelitian yang dilakukan pada penelitian ini adalah menggunakan sebuah database akademik yang disimulasikan secara localhost dari Sistem Informasi Akademik (SIAKAD) Institut Sains dan Teknologi AKPRIND Yogyakarta, dimana pada SIAKAD melakukan proses kegiatan akademik mahasiswa tiap semesternya. Bagian yang akan di teliti adalah bagian proses pengisian krs dan proses pengisian nilai.

Kebutuhan Sistem

Bahan dan alat yang dibutuhkan untuk penelitian ini meliputi hardware dan software, diantaranya yaitu:

- Hardware, laptop dengan Processor Intel(R) Celeron(R) N4120 CPU @ 1.10GHz, 1101 Mhz, 4 Core(s), 4 Logical Processor(s), RAM 8,00 GB, Storage WDC PC SN530 SDBPNPZ-256G-1006.
- Software Sistem Window 11
- Software XAMPP for Windows 8.1.6
- Apache Web Server 2.4.53 PHP/8.1.6
- Database MariaDB 10.4.24

Metode Pengumpulan Data

Metode yang digunakan dalam pengumpulan data pada penelitian ini terdiri dari beberapa metode, yaitu:

Metode Observasi, Metode observasi ini digunakan untuk pengumpulan data dengan pengamatan secara langsung maupun tidak langsung terhadap obyek yang diteliti. Metode Studi Kepustakaan, Metode studi kepustakaan merupakan sebuah cara dalam pengumpulan data dengan mempelajari bahan pustaka baik berupa dokumen tertulis ataupun berupa gambar dengan membandingkan beberapa referensi.

Metode Eksperimen, Metode ini digunakan dengan mengadakan uji coba dan simulasi yang telah dibuat menggunakan RDBMS MariaDB, menguji secara langsung melalui terminal (SQL Manipulation) maupun menggunakan aplikasi berbasis web menggunakan PHP.

Perancangan Sistem

Peraturan Bisnis

Peraturan bisnis pada obyek penelitian adalah bahwa semua proses akademik khususnya bagian isi krs dan isi nilai harus dilakukan di RDBMS dan menggunakan aturan dasar bahwa tiap mahasiswa mengisi KRS akan mendapatkan IDKRS, tiap matakuliah diambil oleh mahasiswa maka daya tampung matakuliah akan berkurang 1 sampai dengan batas daya tampung sama dengan 0 sehingga matakuliah tersebut sudah dianggap penuh. Begitu juga pada sisi mahasiswa, tiap mahasiswa mengambil matakuliah maka pada KRS akan bertambah besar SKS yang diambil sebesar SKS matakuliah tersebut sampai batas SKS yang diperbolehkan untuk mahasiswa tersebut habis (Maksimal SKS yang boleh diambil). Saat proses pengisian nilai, maka hanya bisa mengisikan nilai kepada mahasiswa yang telah melakukan pengisian KRS matakuliah tersebut. Proses transaksi dilakukan menggunakan simulasi aplikasi sederhana menggunakan PHP, dengan menerapkan konsep Store Procedure, Function, View, dan Trigger.

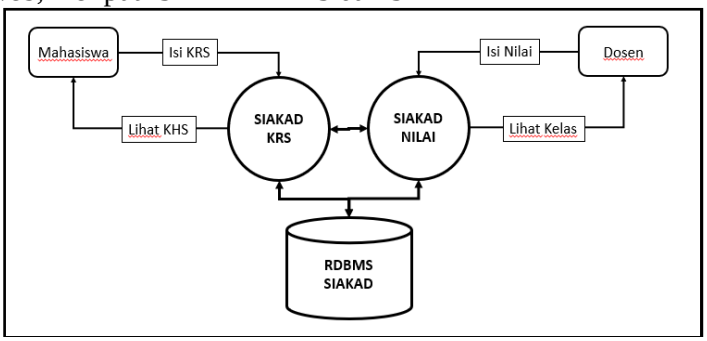
Desain Sistem

Berdasarkan aturan diatas, maka dalam penelitian ini di implementasikan dalam sebuah gambar desain seperti yang ditunjukkan pada gambar 1.

Dalam penelitian ini akan dibangun dan disimulasikan:

1. RDBMS SIAKAD, meliputi tabel-tabel relasional, view, trigger, function dan store procedure

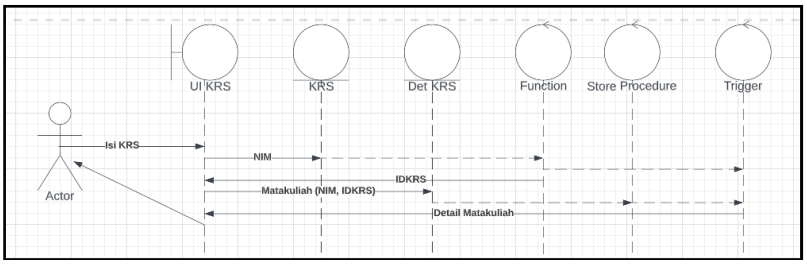
2. Aplikasi web, meliputi SIAKAD KRS dan SIAKAD NILAI



Gambar 1 Desain Sistem

Sequence Diagram Pengisian KRS

Sequence Diagram Pengisian KRS pada gambar 2 digunakan untuk lebih menjelaskan proses yang terjadi saat pengisian KRS.



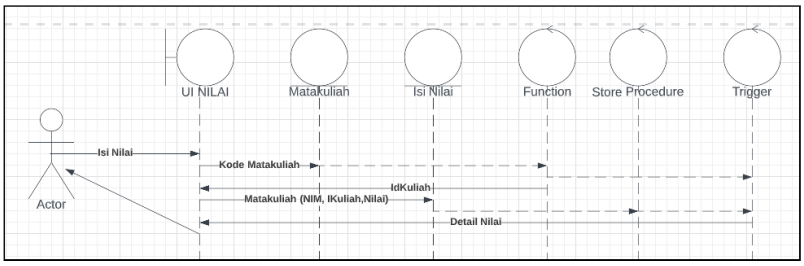
Gambar 2 Sequence Diagram Pengisian KRS

Aktivitas yang terjadi pada proses pengisian KRS dapat dijelaskan sebagai berikut:

1. Aktor (Mahasiswa) melakukan transaksi di UI dengan mengirimkan ke proses KRS berupa NIM, oleh KRS akan diproses melalui function maka akan diperoleh nilai balik IDKRS.
2. Setelah mendapatkan IDKRS, maka dilakukan proses pemilihan matakuliah dengan memanggil store procedure getIsiKRS yang akan menampilkan matakuliah yang belum diambil, selanjutnya hasil pilihan akan dikirim ke store procedure detKRS dan setelah itu trigger after insert adddetkrs akan berjalan untuk Update data tampung kelas, dan Jumlah SKS, jumlah matakuliah serta sisa kuota sks.
3. Untuk proses penggantian matakuliah, maka akan dilakukan pemanggilan matakuliah melalui store procedure getDetKRS terlebih dahulu untuk kemudian dipilih matakuliah yang akan dibatalkan dan akan di proses oleh store procedure DelDetKRS dan secara otomatis trigger after delete pada deldetkrs akan menangani perubahan data baik itu dari krs maupun matakuliah.

Sequence Diagram Pengisian Nilai

Sequence Diagram pada pengisian nilai pada gambar 3 digunakan untuk lebih menjelaskan proses yang terjadi saat pengisian Nilai oleh Dosen.



Gambar 3 Sequence Diagram Pengisian Nilai

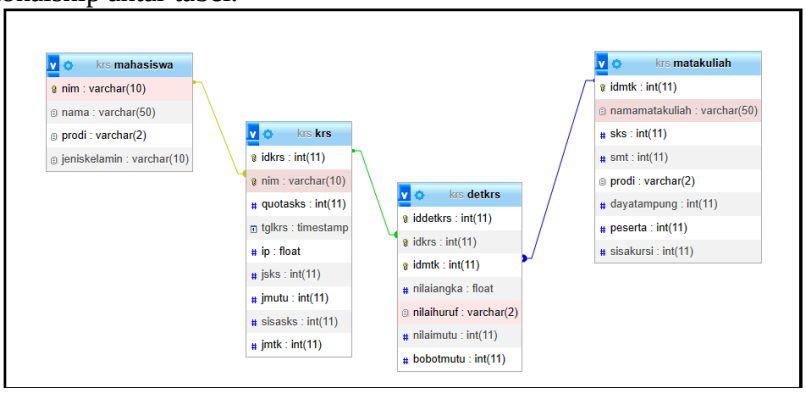
Aktivitas yang terjadi pada proses pengisian Nilai oleh dosen dapat dijelaskan sebagai berikut:

1. Aktor (Dosen) melakukan transaksi di UI dengan view isinilaimatakuliah dan difilter berdasarkan matakuliah yang akan di proses.
2. Setelah mendapatkan IdKuliah, maka dilakukan proses pengisian nilai dengan menampilkan data peserta menggunakan store procedure getIsiNilai, untuk kemudian nilai yang diisikan akan di kirim ke proses store procedure updatenilai, pada proses update ini didukung oleh dua buah function yaitu cNilaiHuruf untuk konversi nilai angka ke nilai huruf dan cNilaiMutu untuk konversi dari nilai angka ke nilai mutu dan juga running trigger updatenilai after update yang akan mengupdate jmutu dan ip pada tabel krs.
3. Untuk proses penggantian nilai matakuliah, maka akan dilakukan seperti point 2.

Perancangan Database

Perancangan Tabel

Tabel-tabel yang digunakan dalam perancangan ini seperti terlihat pada gambar 4 berikut juga tentang relasionalship antar tabel.



Gambar 4 Desain Tabel Relationalship

Tabel Mahasiswa

Tabel ini menampung data mahasiswa dengan primary key nim.

```
CREATE TABLE `mahasiswa` (
  `nim` varchar(10) NOT NULL,
  `nama` varchar(50) NOT NULL,
  `prodi` varchar(2) NOT NULL,
  `jeniskelamin` varchar(10) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COMMENT='Tabel Mahasiswa';
-- Indexes for table `mahasiswa`
--
ALTER TABLE `mahasiswa`
  ADD PRIMARY KEY (`nim`),
  ADD KEY `prodi` (`prodi`);
```

Tabel Matakuliah

Tabel ini menampung data-data matakuliah meliputi idmtk, namamatakuliah, dan juga dayatampung kelas, peserta yang ambil serta sisa kursi yang diberi function check harus lebih atau sama dengan 0. Tabel ini memiliki primary key idmtk, dan index tamu pada field prodi.

```
CREATE TABLE `matakuliah` (
  `idmtk` int(11) NOT NULL,
  `namamatakuliah` varchar(50) NOT NULL,
  `sks` int(11) NOT NULL,
  `smt` int(11) NOT NULL,
  `prodi` varchar(2) NOT NULL,
  `dayatampung` int(11) NOT NULL DEFAULT 40,
  `peserta` int(11) NOT NULL DEFAULT 0,
  `sisakursi` int(11) NOT NULL DEFAULT 40 CHECK (`sisakursi` >= 0)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
-- Indexes for table `matakuliah`
```

```
--
ALTER TABLE `matakuliah`
  ADD PRIMARY KEY (`idmtk`),
  ADD KEY `prodi` (`prodi`);
```

Tabel KRS

Tabel ini menampung data-data KRS berikut jumlah mutu, jumlah sks, sisasks belum diambil, jumlah matakuliah, quota sks dan juga ip semester. Tabel ini memiliki primary index idkrs jenis autoincrement dan index unique pada field nim yang berelasi ke tabel mahasiswa.

```
CREATE TABLE `krs` (
  `idkrs` int(11) NOT NULL,
  `nim` varchar(10) NOT NULL,
  `quotasks` int(11) NOT NULL DEFAULT 24,
  `tglkrs` timestamp NOT NULL DEFAULT current_timestamp(),
  `ip` float NOT NULL DEFAULT 0,
  `jsks` int(11) NOT NULL DEFAULT 0,
  `jmutu` int(11) NOT NULL DEFAULT 0,
  `sisasks` int(11) NOT NULL DEFAULT 24 CHECK (`sisasks` >= 0),
  `jmtk` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- RELATIONSHIPS FOR TABLE `krs`:
--   `nim`
--       `mahasiswa` -> `nim`
--
-- Indexes for table `krs`
--
ALTER TABLE `krs`
  ADD PRIMARY KEY (`idkrs`),
  ADD UNIQUE KEY `nim` (`nim`);
-- AUTO_INCREMENT for table `krs`
--
ALTER TABLE `krs`
  MODIFY `idkrs` int(11) NOT NULL AUTO_INCREMENT;
```

Tabel DetKRS

Tabel ini menampung data matakuliah dan nilai yang diambil oleh seorang mahasiswa, tabel ini menampung perolehan nilai angka, yang secara otomatis akan dihitung pada nilaihuruf melalui cNilaiHuruf dan nilaiangka melalui cNilaiMutu

```
CREATE TABLE `detkrs` (
  `iddetkrs` int(11) NOT NULL,
  `idkrs` int(11) NOT NULL,
  `idmtk` int(11) NOT NULL,
  `nilaiangka` float NOT NULL,
  `nilaihuruf` varchar(2) NOT NULL,
  `nilaimutu` int(11) NOT NULL DEFAULT 0,
  `bobotmutu` int(11) NOT NULL DEFAULT 0
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
--
-- RELATIONSHIPS FOR TABLE `detkrs`:
--   `idkrs`
--       `krs` -> `idkrs`
--   `idmtk`
--       `matakuliah` -> `idmtk`
--
-- Indexes for table `detkrs`
--
ALTER TABLE `detkrs`
  ADD PRIMARY KEY (`iddetkrs`),
  ADD UNIQUE KEY `idkrs_2` (`idkrs`,`idmtk`),
```

```
  ADD KEY `idmtk` (`idmtk`),
  ADD KEY `idkrs` (`idkrs`);
-- AUTO_INCREMENT for table `detkrs`
--
ALTER TABLE `detkrs`
  MODIFY `iddetkrs` int(11) NOT NULL AUTO_INCREMENT;
-- Constraints for table `detkrs`
--
ALTER TABLE `detkrs`
  ADD CONSTRAINT `detkrs_ibfk_3` FOREIGN KEY (`idkrs`) REFERENCES `krs`
(`idkrs`) ON DELETE CASCADE,
  ADD CONSTRAINT `detkrs_ibfk_4` FOREIGN KEY (`idmtk`) REFERENCES `matakuliah`
(`idmtk`);
```

Perancangan Function

Function digunakan untuk mengerjakan sebuah proses dan memberikan nilai balik yang akan digunakan oleh proses yang lain. [7]

Function NewKRS

Function ini digunakan untuk mendaftar KRS pertama kali, dimana function ini akan menerima parameter VNIM yang akan di insertkan ke tabel KRS dan memberikan nilai balik berupa IDKRS.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `NewKRS` (`vnim` VARCHAR(10))
RETURNS INT(11)
BEGIN
DECLARE idkrs int;
INSERT INTO krs(nim) VALUES(vnim);
   SET idkrs = LAST_INSERT_ID();
   RETURN idkrs;
END
```

Function cNilaiHuruf

Function ini digunakan untuk melakukan konversi nilai angka ke nilai huruf, function ini menerima parameter NILAIANGKA dan memberikan nilai balik berupa NILAIHURUF.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `cNilaiHuruf` (`vnilai` INT)
RETURNS VARCHAR(2) CHARSET utf8mb4 DETERMINISTIC NO SQL
BEGIN
DECLARE huruf varchar(2);
IF vnilai > 85 THEN
    set huruf='A';
ELSEIF vnilai > 70 THEN
    set huruf='B';
ELSEIF vnilai > 55 THEN
    set huruf='C';
ELSEIF vnilai > 40 THEN
    set huruf='D';
ELSE
    set huruf='E';
END IF;
   RETURN huruf;
END
```

Function cNilaiMutu

Function ini digunakan untuk melakukan konversi dan menghitung dari nilai angka ke nilai mutu skala 4. Function ini menerima parameter NILAIANGKA dan memberikan balik berupa NILAIMUTU.

```
CREATE DEFINER=`root`@`localhost` FUNCTION `cNilaiMutu` (`vnilai` INT) RETURNS
INT(1) NO SQL
BEGIN
```

```
DECLARE mutu int;
IF vnilai > 85 THEN
    set mutu=4;
ELSEIF vnilai > 70 THEN
    set mutu=3;
ELSEIF vnilai > 55 THEN
    set mutu=2;
ELSEIF vnilai > 40 THEN
    set mutu=1;
ELSE
    set mutu=0;
END IF;
   RETURN mutu;
END
```

Perancangan Store Procedure

Store Procedure digunakan untuk melakukan proses transaksional dan memanggil obyek-obyek lain agar bisa menyelesaikan permasalahan transaksi secara lengkap. [8]

Store Procedure getKRS

Store Procedure digunakan untuk menampilkan data dari tabel KRS dengan parameter NIM, sehingga akan menampilkan 1 record KRS untuk NIM tersebut jika ada datanya.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getKRS` (IN `vnim` VARCHAR(10))
BEGIN
select * from krs where krs.nim=vnim;
END
```

Store Procedure getIsiKRS

Procedure ini akan menampilkan data matakuliah yang belum di ambil oleh seorang mahasiswa, dengan menerima parameter NIM dan SMT.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getIsiKRS` (IN `vnim`
VARCHAR(10), IN `vsmt` INT)
BEGIN
select matakuliah.idmtk, matakuliah.namamatakuliah, matakuliah.sks,
matakuliah.smt, matakuliah.prodi from matakuliah
where matakuliah.smt=vsmt and matakuliah.prodi IN (select prodi from mahasiswa
where nim=vnim) and matakuliah.idmtk not in (SELECT idmtk from detkrs, krs
where detkrs.idkrs=krs.idkrs and krs.nim=vnim);
END
```

Store Procedure DetKRS

Store Procedure ini dikerjakan untuk menyimpan data setelah dilakukan pemilihan matakuliah yang diambil, dengan parameter IDKRS dan IDMTK.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `DetKRS` (`idkrs` INT, `idmtk`
INT)  DETERMINISTIC
BEGIN
   INSERT INTO detkrs(idkrs,idmtk) VALUES(idkrs,idmtk);
END
```

Store Procedure getDetKRS

Store Procedure ini mengirimkan parameter berupa NIM untuk digunakan sebagai filter pada proses ini, sehingga akan menampilkan data matakuliah dan nilai yang telah diambil oleh seseorang.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getDetKRS` (IN `vnim`
VARCHAR(10))  DETERMINISTIC
```

```
BEGIN
select krs.idkrs, detkrs.iddetkrs,matakuliah.idmtk, matakuliah.namamatakuliah,
matakuliah.sks, matakuliah.smt, matakuliah.prodi, detkrs.nilaiangka,
detkrs.nilaihuruf, detkrs.nilaimutu, detkrs.bobotmutu from detkrs, matakuliah
, krs where matakuliah.idmtk=detkrs.idmtk and detkrs.idkrs=krs.idkrs AND
krs.nim=vnim order by matakuliah.smt, matakuliah.idmtk;
END
```

Store Procedure DelDetKRS

Store procedure ini digunakan untuk melakukan penghapusan data detail krs yang telah dipilih oleh tampilan sebelumnya dengan parameter iddetkrs.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `DelDetKRS` (IN `viddetkrs` INT)
BEGIN
   delete from detkrs where iddetkrs=viddetkrs;
END
```

Store Procedure getIsiNilai

Store procedure ini digunakan untuk menampilkan data peserta matakuliah dengan parameter IDMTK, sehingga yang muncul adalah matakuliah yang telah dipilih IDMTKnya.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `getIsiNilai` (IN `vidmtk` INT)
BEGIN
select krs.nim, krs.idkrs,detkrs.iddetkrs, detkrs.idmtk,
mahasiswa.nama, detkrs.nilaiangka
from detkrs,krs,mahasiswa where detkrs.idmtk=vidmtk AND
detkrs.idkrs=krs.idkrs and krs.nim = mahasiswa.nim
order by mahasiswa.nim;
END
```

Store Procedure updatenilai

Store procedure ini digunakan untuk mengupdate nilai angka serta mengkonversi nilai huruf, nilai mutu dan bobot mutu melalui function. Store procedure ini menerima dua parameter yaitu iddetkrs dan nilaiangka.

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `updatenilai` (IN `viddetkrs` INT,
IN `vnilaiangka` INT)
BEGIN
UPDATE detkrs SET nilaiangka=vnilaiangka, nilaihuruf=cNilaiHuruf(vnilaiangka),
nilaimutu=cNilaiMutu(vnilaiangka),
bobotmutu=nilaimutu*(select sks from matakuliah where
matakuliah.idmtk=detkrs.idmtk) WHERE iddetkrs=viddetkrs;
END
```

Perancangan Trigger

Trigger digunakan untuk melakukan proses setelah mendapat pemicu berupa Insert, Update dan Deleted baik itu before atau after. [9]

Trigger adddetkrs

Trigger ini bekerja pada tabel detkrs after insert, setelah tabel detkrs mengalami penambahan data, maka akan melakukan update pada tabel krs terkait jumlah sks, jumlah matakuliah dan sisa sks, juga melakukan update pada tabel matakuliah terkait jumlah peserta dan sisa kursi.

```
CREATE TRIGGER `adddetkrs` AFTER INSERT ON `detkrs` FOR EACH ROW BEGIN
UPDATE krs
set jsks=jsks+(select sks from matakuliah where idmtk=new.idmtk),
jmtk=jmtk+1,sisasks=quotasks-jsks
WHERE idkrs=new.idkrs;
UPDATE matakuliah set peserta=peserta+1,
sisakursi=dayatampung-peserta where idmtk=new.idmtk;
```

```
END
```

Trigger deldetkrs

Trigger ini bekerja pada tabel detkrs after delete, setelah tabel detkrs mengalami penghapusan record, maka trigger ini akan mengupdate tabel krs terkait jumlah sks, jumlah matakuliah, jumlah mutu dan juga ip. Juga pada tabel matakuliah terkait dengan jumlah peserta dan sisa kursi.

```
CREATE TRIGGER `deldetkrs` AFTER DELETE ON `detkrs` FOR EACH ROW BEGIN
UPDATE krs
set jsks=jsks-(select sks from matakuliah where idmtk=old.idmtk),
jmtk=jmtk-1,sisasks=quotasks-jsks,
jmutu=(select sum(bobotmutu) from detkrs where idkrs=krs.idkrs), ip=jmutu/jsks
WHERE idkrs=old.idkrs;
UPDATE matakuliah set peserta=peserta-1,
sisakursi=dayatampung-peserta where idmtk=old.idmtk;
END
```

Trigger updatenilai

Trigger ini bekerja pada tabel detkrs after update, setelah tabel detkrs mengalami perubahan dalam hal ini adalah pengisian/pengubahan nilai maka akan memicu untuk melakukan perubahan pada tabel krs terkait dengan jumlah mutu dan ip.

```
CREATE TRIGGER `updatenilai` AFTER UPDATE ON `detkrs` FOR EACH ROW BEGIN
 update krs set jmutu=(select sum(bobotmutu) from detkrs where
idkrs=krs.idkrs), ip=jmutu/jsks;
 END
```

Perancangan VIEW

View digunakan untuk menampilkan tabel secara langsung sehingga algoritma relationalship yang cukup panjang akan disederhanakan oleh view. [10]

View mahasiswakrs

View ini menampilkan data mahasiswa yang telah mengisi krs, berikut data-data hasil krsnya.

```
CREATE VIEW `mahasiswakrs`  AS SELECT `mahasiswa`.`nim` AS `nim`,
`mahasiswa`.`nama` AS `nama`, `mahasiswa`.`prodi` AS `prodi`, `idkrs` AS
`idkrs`, `tglkrs` AS `tglkrs`, `jsks` AS `jsks`, `jmtk` AS `jmtk`, `sisasks`
AS `sisasks`, `quotasks` AS `quotasks`, `jmutu` AS `jmutu`, `ip` AS `ip` FROM
(`mahasiswa` join `krs`) WHERE `mahasiswa`.`nim` = `nim``nim`  ;
```

View pesertamatakuliah

View ini menampilkan data peserta matakuliah yang telah mengisi KRS.

```
CREATE VIEW `pesertamatakuliah`  AS SELECT `nim` AS `nim`, `idkrs` AS `idkrs`,
`detkrs`.`iddetkrs` AS `iddetkrs`, `detkrs`.`idmtk` AS `idmtk`,
`mahasiswa`.`nama` AS `nama`, `detkrs`.`nilaiangka` AS `nilaiangka` FROM
((`detkrs` join `krs`) join `mahasiswa`) WHERE `detkrs`.`idkrs` = `idkrs` AND
`nim` = `mahasiswa`.`nim` ORDER BY `mahasiswa`.`nim` ASC  ;
```

View isinilaimatakuliah

View ini menampilkan data matakuliah yang memiliki peserta lebih dari 0.

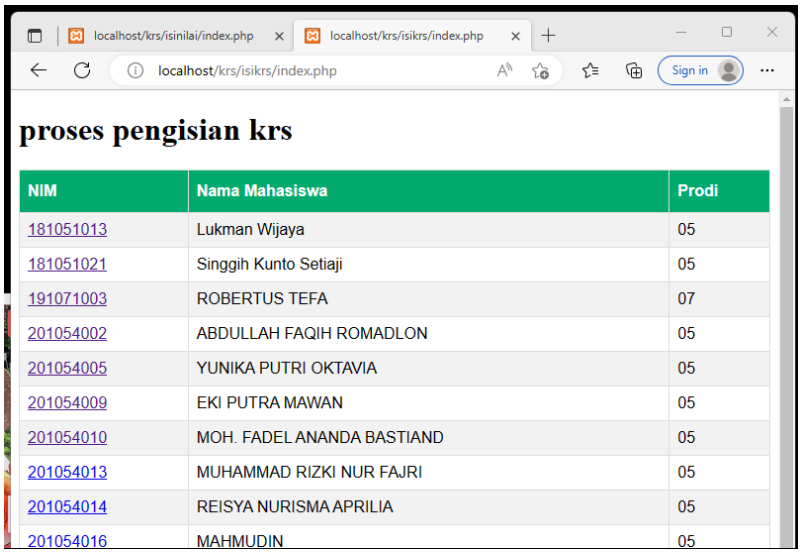
```
CREATE VIEW `isinilaimatakuliah`  AS SELECT `matakuliah`.`idmtk` AS `idmtk`,
`matakuliah`.`namamatakuliah` AS `namamatakuliah`, `matakuliah`.`sks` AS
`sks`, `matakuliah`.`smt` AS `smt`, `matakuliah`.`prodi` AS `prodi`,
`matakuliah`.`peserta` AS `peserta` FROM `matakuliah` WHERE
`matakuliah`.`peserta` > 0  ;
```

3. HASIL DAN PEMBAHASAN

Implementasi dari rancangan berupa sebuah aplikasi yang terbagi menjadi dua kelompok yaitu Pengisian KRS dan Pengisian Nilai.

Pengisian KRS

Saat awal pertama kali dibuka, maka akan ditampilkan data mahasiswa yang ada, dengan pilihan/button pada kolom NIM untuk melakukan proses pengisian KRS terlihat pada gambar 5.



Gambar 5 Proses Pengisian KRS

Untuk membangkitkan tampilan tersebut menggunakan script:

```
$sql = "SELECT nim,nama,prodi FROM mahasiswa";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
  ?>
  <table id="krs"><th>NIM</th><th>Nama Mahasiswa</th><th>Prodi</th>
  <?php
  while($row = $result->fetch_assoc()) {
    echo "<tr><td><a href=isikrs.php?vnim=" . $row["nim"]. ">" . $row["nim"].
"</a>
```

Url dari tampilan tersebut adalah ke isikrs.php dengan membawa parameter nim dari mahasiswa yang dipilih, data akan diambilkan dari view mahasiswakrs dengan filter nim seperti pada gambar 6.

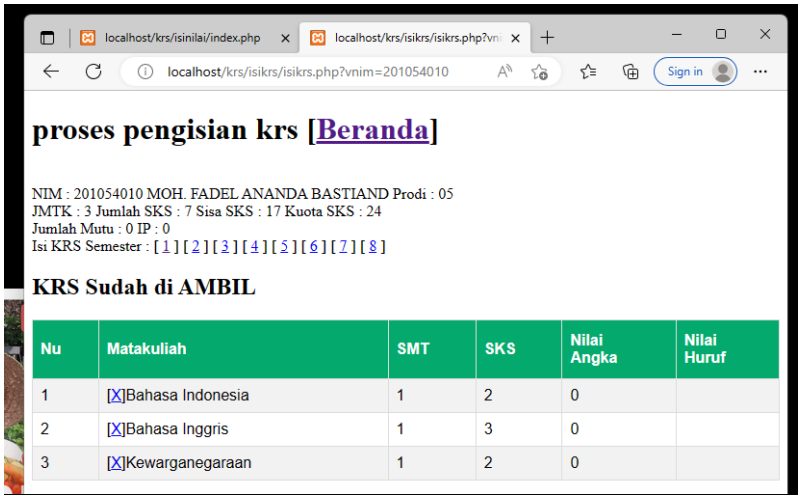
```
$sql = "SELECT * FROM mahasiswakrs where nim='".$_GET['vnim']."'";
$result = $conn->query($sql);
```

Sedangkan jika belum pernah mengisi KRS maka akan dilakukan proses pemanggilan function NewKRS dengan parameter NIM, lalu dipanggil ulang scriptnya dengan header location.

```
  $sql = "SELECT NewKRS(".$_GET['vnim'].")";
  $result = $conn->query($sql);
  header("location:isikrs.php?vnim=".$_GET['vnim']);
```

Setelah itu, data KRS yang telah diisikan ditampilkan dengan memanggil store procedure getDetKRS dengan parameter NIM.

```
//list krs
  $sql1 = "call getDetKRS('".$_GET['vnim']."')";
  $result1 = $conn->query($sql1);
```



Gambar 6 Tampilan Isi KRS

Pada gambar 7 merupakan tampilan dari matakuliah yang belum diambil oleh mahasiswa tersebut pada semester 1. Dari Script isikrs.php pada bagian untuk memilih semester yang akan di pilih.

```
echo "<span><br>Isi KRS Semester : ";
  for($i=1;$i<9;$i++) echo "[ <a
href=addisidetkrs.php?vnim=".$_GET['vnim']."&vsmt=".$i."&vidkrs=".$row['idkrs'
].">".$i."</a> ] ";
```

Sehingga pada script addisidetkrs.php dengan membawa parameter NIM dan SMT, akan di panggilkan store procedure getIsiKRS untuk menampilkan tampilan pada gambar 7.

```
 $sql2 = "call getIsiKRS('".$_GET['vnim']."',".$_GET['vsmt'].")";
    $result2 = $conn->query($sql2);
```

Pada script tersebut ada href / link untuk memilih matakuliah yang akan diambil dengan membawa parameter NIM, IDKRS dan IDMTK yang dipilih untuk di kirim ke script isidetkrs.php.

```
while($row2 = $result2->fetch_assoc()) {
    echo "<tr><td><a href=isidetkrs.php?vnim=".$_GET["vnim"] ."&vidkrs=".
$_GET["vidkrs"] ."&vidmtk=". $row2["idmtk"]. "&vsmt=".$row2["smt"].">"
```

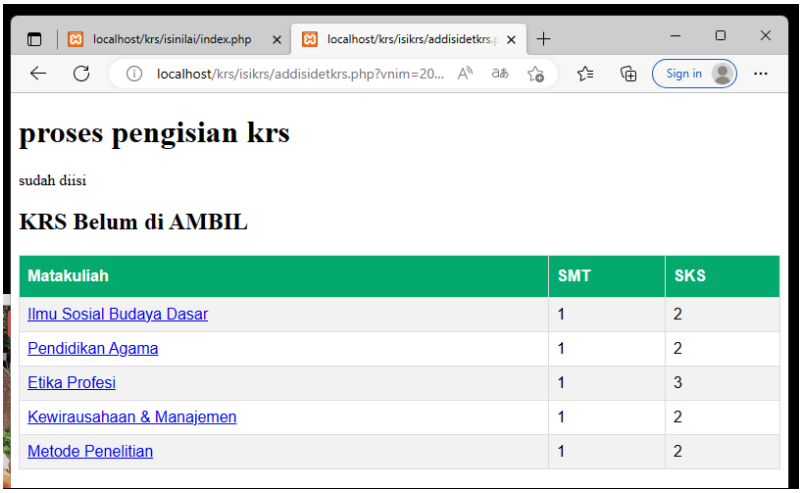
Sehingga pada script isidetkrs, akan diproses dengan memanggil store procedure DetKRS dengan mengirimkan parameter IDKRS dan IDMTK.

```
 $sql = "call DetKRS(".$_GET['vidkrs'].",".$_GET['vidmtk'].")";
  if($conn->query($sql))
  header("location:isikrs.php?vnim=".$_GET['vnim']);
```

Pembatalan Matakuliah

Sedangkan untuk pembatalan matakuliah yang telah diambil pada gambar 6 ada tombol **X**, maka akan mengirim ke script bataldetkrs.php dengan parameter NIM dan IDDETKRS. Untuk selanjutnya proses pembatalan dengan memanggil store procedure DelDetKRS dengan parameter IDDETKRS.

```
 $sql = "call DelDetKRS(".$_GET['viddetkrs'].")";
  $result=$conn->query($sql);
```

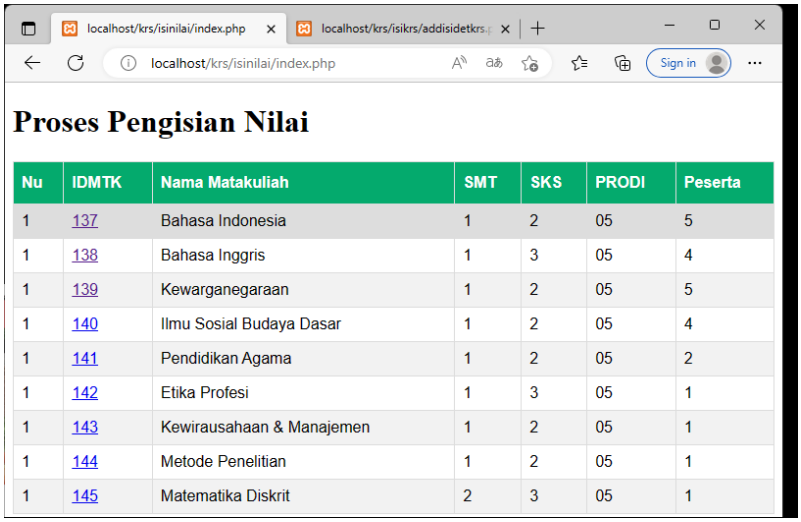


Gambar 7 Proses pemilihan matakuliah

Proses Pengisian Nilai

Proses pengisian nilai akan diawali dengan penampilan data matakuliah yang sudah ada pesertanya seperti pada gambar 8, data dibangkitkan dengan memanggil view isinilaimatakuliah, pada halaman ini disediakan href/url untuk mengisikan nilai pada isinilai.php dengan parameter idmtk. Seperti pada script berikut:

```
$sql = "SELECT * FROM isinilaimatakuliah";
$result = $conn->query($sql);
```



Gambar 8 Halaman Depan Pengisian Nilai

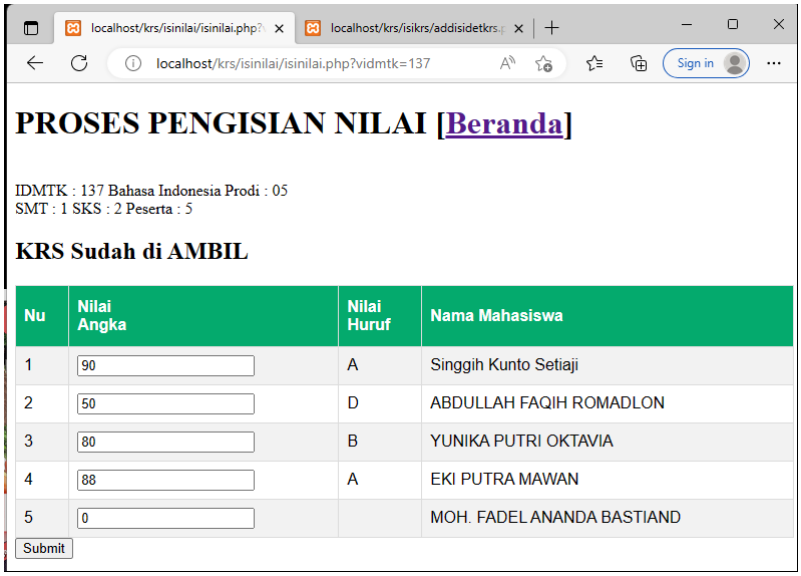
Pada script isinilai.php, akan di panggil store procedure getIsiNilai dengan parameter IDMTK, sehingga akan ditampilkan data peserta dari matakuliah tersebut seperti pada gambar 9

```
//list peserta
  $sql1 = "call getIsiNilai('".$_GET['vidmtk']."')";
  $result1 = $conn->query($sql1);
```

Pada proses ini dibuat sebuah form untuk menampung data IDMTK dan array dari IDDETKRS serta NILAIANGKA yang diisikan. Untuk kemudian di proses oleh simpanisinilai.php seperti script berikut, dengan pengelolaan loop/array data yang ada dan di panggil store procedure untuk menyimpan/update nilai angka dengan store procedure updatenilai dengan parameter IDDETKRS dan NILAI ANGKA yang diisikan.

```
for($i=1;$i<=count($_POST['iddetkrs']);$i++)
{
```

```
echo "<br>$i=".$_POST['iddetkrs'][$i];
$sql = "call updatenilai(".$_POST['iddetkrs'][$i].",".$_POST['nilai'][$i].")";
$result = $conn->query($sql);
}
```



Gambar 11 Halaman Pengisian Nilai

Dari implementasi di aplikasi terlihat bahwa dengan menggunakan metode perancangan ini, maka perintah SQL yang komplek tidak pernah muncul atau digunakan, karena sudah dikelola langsung pada sisi RDBMS, begitu juga jika terjadi kesalahan algoritma dalam aturan bisnis tidak akan terjadi pada sisi aplikasi.

4. KESIMPULAN

Dari hasil penelitian yang telah dilakukan dapat disimpulkan bahwa dengan pengelolaan algoritma database pada sisi RDBMS maka kesalahan dalam menginterprestasikan diagram relasionalship serta keterkaitan antar tabel dengan tabel yang lain akan minimal, begitu juga dengan rule bisnis aplikasi akan lebih terjamin karena semua dikelola pada RDBMS. Dengan penerapan Store Procedure, Function, Trigger serta View yang tepat maka akan dihasilkan sebuah rancangan database yang kuat secara algoritma. Dengan metode ini, maka pada bagian pengembang aplikasi akan menjadi sangat mudah dan ringan karena tidak dibebani oleh diagram relasionalship maupun algoritma-algoritma dari rule bisnis aplikasi, cukup dengan memanggil function, store procedure maupun view yang telah disediakan.

5. SARAN

Pada penelitian ini belum dibahas tentang kecepatan proses / respon time jika dibandingkan dengan metode konvensional. Sehingga perlu dilakukan penelitian lanjutan apakah respon time dengan metode ini menjadi lebih bagus atau lebih jelek.

DAFTAR PUSTAKA

[1] J. R. W. P. N. &. O. A. J. Groff, SQL The Complete Refference - Third Edition, United States: The McGraw-Hill Companies, 2010.

[2] J. Triyono, "KONSEP MEMBANGUN APLIKASI MULTIPLATFORM DENGAN OPTIMALISASI PENGGUNAAN VIEW, FUNCTION DAN TRIGGER PADA RDBMS POSTGRESQL," *in Simposium Nasional RAPI XV – 2016 FT UMS*, Surakarta, 2016.

[3] Z. Alfiani, "Perancangan Sistem Informasi Kerja Praktik dan Tugas Akhir pada Program Studi Teknik Industri Universitas Bandung," in *Prosiding Teknik Industri*, Bandung, 2017.

[4] J. Triyono, P. Nadira and C. A. Subhkan, "IMPLEMENTASI SISTEM TERDISTRIBUSI MENGGUNAKAN REPLIKASI DATABASE DAN WEB SERVICE," *in Prosiding Seminar Nasional Multidisiplin Ilmu*, Yogyakarta, 2021.

[5] E. Fatkhiyah, J. Triyono and E. N. Cahyo, "PERANCANGAN DATABASE PENDAFTARAN SWAB ANTIGEN BERBASIS ONLINE," in *Prosiding Seminar Nasional Multidisiplin Ilmu*, Yogyakarta, 2021.

[6] J. Triyono, P. Haryani and A. Padmanaba, "Model Kontrol Transaksi RDBMS Menggunakan Trigger dan Waktu Server," *Jurnal Teknologi*, vol. 12, no. 1, pp. 80-86, 2019.

[7] java T point, "MariaDB Functions," Java T Point, [Online]. Available: https://www.javatpoint.com/mariadb-function. [Accessed 12 Desember 2022].

[8] javaTpoint, "MariaDB Procedure," javaTpoint, [Online]. Available: https://www.javatpoint.com/mariadb-procedure. [Accessed 12 Desember 2022].

[9] MySQLCode, "Introduction to MySQL Triggers – Definition, Types, and Syntax," MySQLCode, [Online]. Available: https://mysqlcode.com/mysql-triggers-introduction/. [Accessed 24 Desember 2022].

[10] MySQL Tutorial, "MySQL Views," MySQL Tutorial, [Online]. Available: https://www.mysqltutorial.org/mysql-views-tutorial.aspx. [Accessed 15 Desember 2022].