SISTEM MANAJEMEN BASIS DATA
IMPLEMENTASI TRIGGER DAN CHECK FUNCTION
STUDI KASUS PEMESANAN TIKET BIOSKOP

Disusun oleh:

Nama : Nur Rohman Ardani

NIM : 14.52.0540

Nama : Tusmiyati

NIM : 14.52.0548

Nama : Adhika Pramita

NIM : 14.52.0530

PROGRAM MAGISTER TEKNIK INFORMATIKA

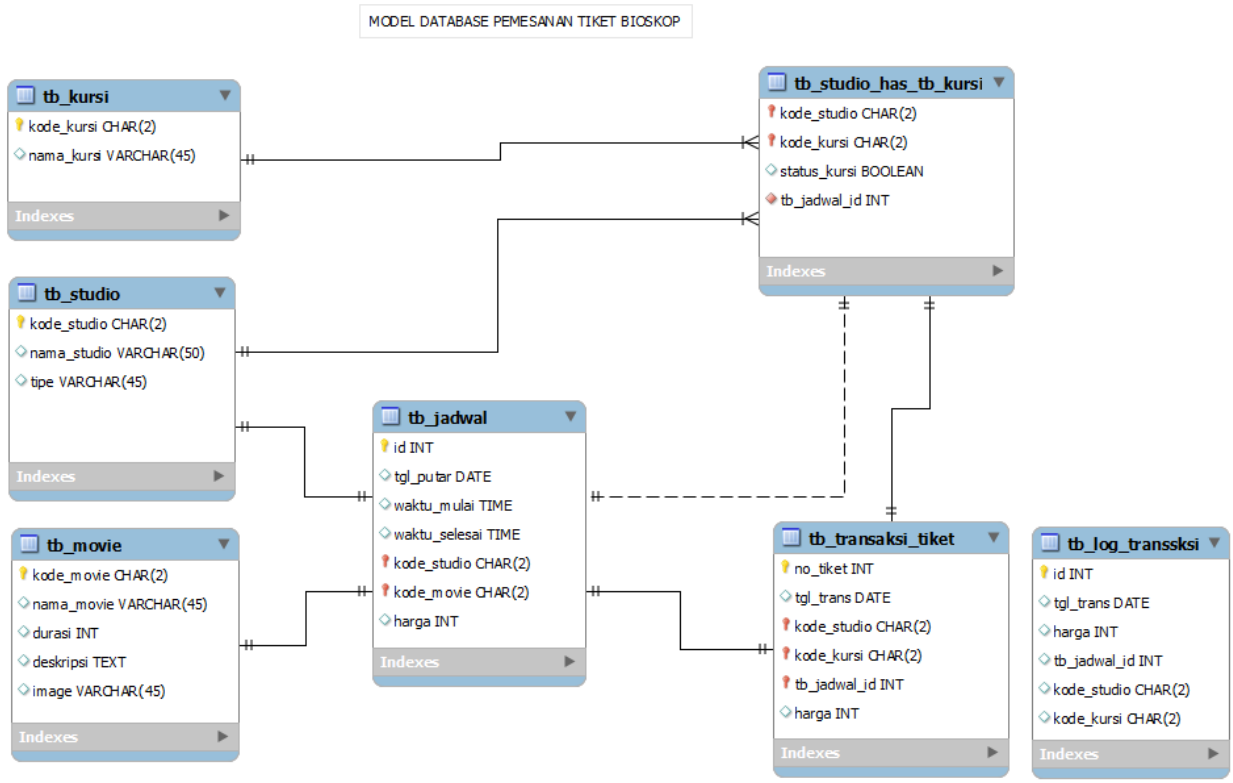
PROGRAM PASCASARJANA STMIK AMIKOM YOGYAKARTA

YOGYAKARTA

2014

I. DESAIN MODEL DATABASE

Sebelum masuk ke dalam implementasi dan pembahasan berikut rancangan database pemesanan tiket bioskop.



Gambar 1 Pemodelan Database Pemesanan Tiket Bioskop

Penggunaan implementasi dari trigger dan check function pada database ***POSTGRESQL 64 BIT*** pada windows 64 Bit

II. SCRIPT SQL

Create Database : **bdbioskop**

```
drop database if exists dbbioskop;
drop role if exists dbbioskop;

create user smbd createdb createuser password 'smbd';
create database dbbioskop owner smbd;
```

Create Table :

```
-- -----------------------------------------------------
-- Table `tb_kursi`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `tb_kursi` ;

CREATE TABLE IF NOT EXISTS `tb_kursi` (
  `kode_kursi` CHAR(2) NOT NULL,
  `nama_kursi` VARCHAR(45) NULL,
  PRIMARY KEY (`kode_kursi`));

-- -----------------------------------------------------
-- Table `tb_studio`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `tb_studio` ;

CREATE TABLE IF NOT EXISTS `tb_studio` (
  `kode_studio` CHAR(2) NOT NULL,
  `nama_studio` VARCHAR(50) NULL,
  `tipe` VARCHAR(45) NULL,
  PRIMARY KEY (`kode_studio));

-- -----------------------------------------------------
-- Table `tb_movie`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `tb_movie` ;

CREATE TABLE IF NOT EXISTS `tb_movie` (
  `kode_movie` CHAR(2) NOT NULL,
  `nama_movie` VARCHAR(45) NULL,
  `durasi` Int NULL,
  `deskripsi` TEXT NULL,
  `image` VARCHAR(45) NULL,
  PRIMARY KEY (`kode_movie`));


-- -----------------------------------------------------
-- Table `tb_jadwal`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `tb_jadwal` ;

CREATE TABLE IF NOT EXISTS `tb_jadwal` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `tgl_putar` DATE NULL,
  `waktu_mulai` TIME NULL,
  `waktu_selesai` TIME NULL,
  `kode_studio` CHAR NOT NULL,
  `kode_movie` CHAR NOT NULL,
  `harga` INT NULL,
  PRIMARY KEY (`id`, `kode_studio`, `kode_movie`),
  CONSTRAINT `fk_kode_studio`
    FOREIGN KEY (`kode_studio`)
    REFERENCES `tb_studio` (`kode_studio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_kode_movie`
    FOREIGN KEY (`kode_movie`)
    REFERENCES `tb_movie` (`kode_movie`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

```
-- -----------------------------------------------------
-- Table `tb_studio_has_tb_kursi`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `tb_studio_has_tb_kursi` ;

CREATE TABLE IF NOT EXISTS `tb_studio_has_tb_kursi` (
  `kode_studio` CHAR(2) NOT NULL,
  `kode_kursi`  CHAR(2) NOT NULL,
  `status_kursi` TINYINT(1) NULL,
  `tb_jadwal_id` INT NOT NULL,
  PRIMARY KEY (`kode_studio`, `kode_kursi`),
  INDEX `fk_tb_studio_has_tb_kursi_tb_kursi1_idx` (`kode_kursi` ASC),
  INDEX `fk_tb_studio_has_tb_kursi_tb_studio_idx` (`kode_studio` ASC),
  CONSTRAINT `fk_tb_studio_has_tb_kursi_tb_studio`
    FOREIGN KEY (`kode_studio`)
    REFERENCES `tb_studio` (`kode_studio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_tb_studio_has_tb_kursi_tb_jadwal1`
    FOREIGN KEY (`tb_jadwal_id`)
    REFERENCES `tb_jadwal` (`id`)
  CONSTRAINT `fk_tb_studio_has_tb_kursi_tb_kursi1`
    FOREIGN KEY (`kode_kursi`)
    REFERENCES `tb_kursi` (`kode_kursi`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);

-- -----------------------------------------------------
-- Table `tb_transaksi_tiket`
-- -----------------------------------------------------
DROP TABLE IF EXISTS `tb_transaksi_tiket` ;

CREATE TABLE IF NOT EXISTS `tb_transaksi_tiket` (
  `no_tiket` INT NOT NULL AUTO_INCREMENT,
  `tgl_trans` DATE NULL,
  `kode_studio` CHAR(2) NOT NULL,
  `kode_kursi`  CHAR(2)NOT NULL,
  `tb_jadwal_id` INT NOT NULL,
  PRIMARY KEY (`no_tiket`, `kode_studio`, `kode_kursi`, `tb_jadwal_id`),

  CONSTRAINT `fk_tb_transaksi_tiket_tb_studio_has_tb_kursi1`
    FOREIGN KEY (`kode_studio` , `kode_kursi`)
    REFERENCES `tb_studio_has_tb_kursi` (`kode_studio` , `kode_kursi`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_tb_transaksi_tiket_tb_jadwal1`
    FOREIGN KEY (`tb_jadwal_id`)
    REFERENCES `tb_jadwal` (`id`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

III.Penjelasan Trigger Dan Function

Function atau dikenal store procedure adalah fungsi yang mempunyai nilai balik yang mampu memuat sebuah operasi perhitungan

Struktur Function

```
CREATE [OR REPLACE] FUNCTION function_name (arguments)
RETURNS return_datatype AS $variable_name$
  DECLARE
    declaration;
    [...]
  BEGIN
    < function_body >
    [...]
    RETURN { variable_name | value }
  END; LANGUAGE plpgsql;
```

Contoh Penggunaan Function

```
CREATE OR REPLACE FUNCTION totalRecords ()
RETURNS integer AS $total$
declare
   total integer;
BEGIN
   SELECT count(*) into total FROM COMPANY;
   RETURN total;
END;
$total$ LANGUAGE plpgsql;
```

Trigger adalah fungsi yang berada dalam tabel, dan berjalan secara otomatis jika tabel tersebut dikenai operasi insert, update, delete. Trigger memiliki manfaat

a. Filter
b. Backup
c. Log

Struktur Trigger

```
CREATE [OR REPLACE] FUNCTION function_name ()
RETURNS trigger AS $variable_name$
  DECLARE
    declaration;
    [...]
  BEGIN
    < function_body >
    [...]
    RETURN { variable_name | value }
  END; LANGUAGE plpgsql;
```

Perbedaan hanya pada returns trigger. Jika pada function returns tipe data tapi jika pada trigger returns trigger.

Menambahkan trigger pada tabel

```
CREATE TRIGGER nama_trigger [AFTER,BEFORE] [INSERT,UPDATE,DELETE] ON
(nama_tabel) FOR EACH ROW PROSEDURES  function_trigger();
```

IV. Implementasi Function Dan Trigger

1. Function Status Kursi

Function ini digunakan untuk membatasi jika kursi sudah dipesan maka data pesan kursi tidak bisa dimasukan dan dimunculkan pesan status kursi sudah dipesan

```
CREATE FUNCTION status_kursi (jadwal_id INTEGER,kode_studio
CHAR(2), kode_kursi CHAR(2))
RETURNS BOOLEAN AS
$$
DECLARE
 statuskursi : INTEGER
BEGIN
 SELECT COUNT(*) INTO statuskursi FROM tb_studio_has_tb_kursi WHERE
kode_studio ='$2' AND kode_kursi = '$3' AND tb_jadwal_id = '$1' AND
STATUS = TRUE;
 IF(statuskursi = 1) THEN
   RAISE EXCEPTION "KURSI SUDAH DIPESAN ";
   RETURN FALSE;
  ELSE
   RETURN TRUE;
END
$$ LANGUAGE plpgsql;
```

Implementasi :

```
CREATE TABLE IF NOT EXISTS `tb_transaksi_tiket` (
  `no_tiket` INT NOT NULL AUTO_INCREMENT,
  `tgl_trans` DATE NULL,
  `kode_studio` CHAR(2) NOT NULL,
  `kode_kursi` CHAR(2) NOT NULL,
  `tb_jadwal_id` INT NOT NULL,
  PRIMARY    KEY    (`no_tiket`,    `kode_studio`,    `kode_kursi`,
`tb_jadwal_id`),
  CONSTRAINT `fk_tb_transaksi_tiket_tb_studio_has_tb_kursi1`
    FOREIGN KEY (`kode_studio` , `kode_kursi`)
    REFERENCES     `tb_studio_has_tb_kursi`     (`kode_studio`     ,
`kode_kursi`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_tb_transaksi_tiket_tb_jadwal1`
    FOREIGN KEY (`tb_jadwal_id`)
    REFERENCES `tb_jadwal` (`id`)
    CHECK status_kursi(tb_jadwal_id,kode_studio, kode_kursi)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

2. Function Kursi Tersedia

Function berfungsi untuk menampilkan jumlah kursi yang tersedia pada studio

```
CREATE FUNCTION kursi_tersedia (jadwal_id INTEGER,kode_studio
CHAR(2))
RETURNS INTEGER AS
$$
DECLARE
 statuskursi : INTEGER
BEGIN
 SELECT COUNT(*) INTO statuskursi FROM tb_studio_has_tb_kursi WHERE
tb_jadwal_id= '$1' AND kode_studio ='$2' AND STATUS = FALSE;

   RETURN statuskursi;

END
$$ LANGUAGE plpgsql;
```

3. Function Kursi Terpakai

Function berfungsi untuk menampilkan jumlah kursi yang terpakai pada studio

```
CREATE FUNCTION kursi_terpakai (jadwal_id INTEGER ,kode_studio
CHAR(2))
RETURNS INTEGER AS
$$
DECLARE
 statuskursi : INTEGER
BEGIN
 SELECT COUNT(*) INTO statuskursi FROM tb_studio_has_tb_kursi WHERE
tb_jadwal_id = '$1' AND kode_studio ='$2' AND STATUS = TRUE;

   RETURN statuskursi;

END
$$ LANGUAGE plpgsql;
```

4. Function Kursi Penuh

Function berfungsi untuk membatasi jumlah kursi keseluruhan sudah terpakai maka pemesanan tiket tidak bisa dilakukan

```
CREATE FUNCTION kursi_penuh(jadwal_id INTEGER,kode_studio CHAR(2))
RETURNS BOOLEAN AS
$$
DECLARE
 statuskursi : INTEGER
 jumlah_kursi : INTEGER
 BEGIN
 SELECT COUNT(*) INTO jumlah_kursi FROM tb_kursi

 SELECT COUNT(*) INTO statuskursi FROM tb_studio_has_tb_kursi WHERE
tb_jadwal_id = '$1' AND kode_studio ='$2' and AND STATUS = FALSE;
```

```
IF(statuskursi >= jumlah_kursi) THEN
   RETURN FALSE;
  ELSE
   RETURN TRUE;
END
$$ LANGUAGE plpgsql;
```

Penggunaan :

```
CREATE TABLE IF NOT EXISTS `tb_studio_has_tb_kursi` (
  `kode_studio` CHAR(2) NOT NULL,
  `kode_kursi`  CHAR(2) NOT NULL,
  `status_kursi` TINYINT(1) NULL,
  `tb_jadwal_id` INT NOT NULL,
  PRIMARY KEY (`kode_studio`, `kode_kursi`),
  INDEX `fk_tb_studio_has_tb_kursi_tb_kursi1_idx` (`kode_kursi` ASC),
  INDEX `fk_tb_studio_has_tb_kursi_tb_studio_idx` (`kode_studio` ASC),
  CONSTRAINT `fk_tb_studio_has_tb_kursi_tb_studio`
    FOREIGN KEY (`kode_studio`)
    REFERENCES `tb_studio` (`kode_studio`)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
  CONSTRAINT `fk_tb_studio_has_tb_kursi_tb_jadwal1`
    FOREIGN KEY (`tb_jadwal_id`)
    REFERENCES `tb_jadwal` (`id`)
  CONSTRAINT `fk_tb_studio_has_tb_kursi_tb_kursi1`
    FOREIGN KEY (`kode_kursi`)
    REFERENCES `tb_kursi` (`kode_kursi`)
  CHECK kursi_penuh(tb_jadwal_id,kode_studio)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION);
```

5. Trigger Update Status Kursi

Trigger untuk mengubah status kursi menjadi terpakai (true) jika tiket sudah dibeli atau ditransaksikan.

```
CREATE FUNCTION update_kursi()
RETURNS TRIGGER AS
$$
BEGIN
  UPDATE tb_studio_has_tb_kursi SET status=true where kode_kursi =
NEW.kode_kursi AND kode_studio = NEW.kode_studio AND tb_jadwal_id =
NEW.tb_jadwal_id

END
$$ LANGUAGE plpgsql;
```

Penggunaan :

```
create trigger update_status_kursi AFTER INSERT ON tb_transaksi_tiket
FOR EACH ROW PROSEDURES update_kursi();
```

6. Trigger Studio Mulai

Trigger secara otomatis mempersiapkan kursi dengan status false pada saat pembuatan jadwal

```
CREATE FUNCTION studio_mulai()
RETURNS TRIGGER AS
$$
BEGIN

  INSERT INTO
tb_studio_has_kursi(tb_jadwal_id,kode_kursi,kode_studio) values
(NEW.id,(SELECT kode_kursi,kode_studio FROM tb_kursi));

END
$$ LANGUAGE plpgsql;
```

Penggunaan :

```
create trigger insert_studio_mulai AFTER INSERT ON tb_jadwal FOR EACH
ROW PROSEDURES studio_mulai();
```

7. Trigger Log Transaksi

Trigger untuk menyimpan data transaksi yang telah dihapus.

```
CREATE FUNCTION log_transaksi()
RETURNS TRIGGER AS
$$
BEGIN

INSERT INTO
tb_log_traksaksi(tgl_trans,harga,kode_studio,kode_kursi,jadwal_id,d
eleted_at)
VALUES (OLD.tgl_trans,OLD.harga,OLD.kode_studio,
OLD.kode_kursi,OLD.jadwal_id,NOW())

END
$$ LANGUAGE plpgsql;
```