**Tugas Database, Function, Trigger dan Store Procedure**

Dari tabel-tabel yang telah ternormalisasi pada tugas sebelumnya, dituangkan ke dalam ERD Diagram menggunakan mysql workbench didapatkan dll file berikut ini :

-- MySQL Workbench Forward Engineering

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

-- -----------------------------------------------------

-- Schema programmer\_job

-- -----------------------------------------------------

-- -----------------------------------------------------

-- Schema programmer\_job

-- -----------------------------------------------------

CREATE SCHEMA IF NOT EXISTS `programmer\_job` DEFAULT CHARACTER SET utf8 ;

USE `programmer\_job` ;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_admin`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_admin` (

`id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(255) NOT NULL,

`password` VARCHAR(255) NOT NULL,

`fullname` VARCHAR(255) NOT NULL,

`created\_account` DATETIME NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_privilege`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_privilege` (

`id` INT NOT NULL AUTO\_INCREMENT,

`privilege` VARCHAR(255) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_admin\_privilege`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_admin\_privilege` (

`id` INT NOT NULL AUTO\_INCREMENT,

`tb\_admin\_id` INT NOT NULL,

`tb\_privilege\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_admin\_privilege\_tb\_admin\_idx` (`tb\_admin\_id` ASC) ,

INDEX `fk\_tb\_admin\_privilege\_tb\_privilege1\_idx` (`tb\_privilege\_id` ASC) ,

CONSTRAINT `fk\_tb\_admin\_privilege\_tb\_admin`

FOREIGN KEY (`tb\_admin\_id`)

REFERENCES `programmer\_job`.`tb\_admin` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_admin\_privilege\_tb\_privilege1`

FOREIGN KEY (`tb\_privilege\_id`)

REFERENCES `programmer\_job`.`tb\_privilege` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_address`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_address` (

`postal\_code` VARCHAR(10) NOT NULL,

`district` VARCHAR(255) NOT NULL,

`city` VARCHAR(255) NOT NULL,

`country` VARCHAR(255) NOT NULL,

PRIMARY KEY (`postal\_code`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_software\_programmer`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_software\_programmer` (

`id` INT NOT NULL AUTO\_INCREMENT,

`username` VARCHAR(255) NOT NULL,

`password` VARCHAR(255) NOT NULL,

`email` VARCHAR(255) NOT NULL,

`fullname` VARCHAR(255) NOT NULL,

`created\_account` DATETIME NOT NULL,

`tb\_admin\_id` INT NOT NULL,

`tb\_address\_postal\_code` VARCHAR(10) NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_software\_programmer\_tb\_admin1\_idx` (`tb\_admin\_id` ASC) ,

INDEX `fk\_tb\_software\_programmer\_tb\_address1\_idx` (`tb\_address\_postal\_code` ASC) ,

CONSTRAINT `fk\_tb\_software\_programmer\_tb\_admin1`

FOREIGN KEY (`tb\_admin\_id`)

REFERENCES `programmer\_job`.`tb\_admin` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_software\_programmer\_tb\_address1`

FOREIGN KEY (`tb\_address\_postal\_code`)

REFERENCES `programmer\_job`.`tb\_address` (`postal\_code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_company`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_company` (

`id` INT NOT NULL AUTO\_INCREMENT,

`companyname` VARCHAR(255) NOT NULL,

`description` LONGTEXT NULL,

`tb\_address\_postal\_code` VARCHAR(10) NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_company\_tb\_address1\_idx` (`tb\_address\_postal\_code` ASC) ,

CONSTRAINT `fk\_tb\_company\_tb\_address1`

FOREIGN KEY (`tb\_address\_postal\_code`)

REFERENCES `programmer\_job`.`tb\_address` (`postal\_code`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_company\_sp`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_company\_sp` (

`id` INT NOT NULL AUTO\_INCREMENT,

`tb\_software\_programmer\_id` INT NOT NULL,

`tb\_company\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_company\_sp\_tb\_software\_programmer1\_idx` (`tb\_software\_programmer\_id` ASC) ,

INDEX `fk\_tb\_company\_sp\_tb\_company1\_idx` (`tb\_company\_id` ASC) ,

CONSTRAINT `fk\_tb\_company\_sp\_tb\_software\_programmer1`

FOREIGN KEY (`tb\_software\_programmer\_id`)

REFERENCES `programmer\_job`.`tb\_software\_programmer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_company\_sp\_tb\_company1`

FOREIGN KEY (`tb\_company\_id`)

REFERENCES `programmer\_job`.`tb\_company` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_chatroom`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_chatroom` (

`id` INT NOT NULL AUTO\_INCREMENT,

`roomname` VARCHAR(45) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_room\_member`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_room\_member` (

`id` INT NOT NULL AUTO\_INCREMENT,

`tb\_software\_programmer\_id` INT NOT NULL,

`tb\_chatroom\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_room\_member\_tb\_software\_programmer1\_idx` (`tb\_software\_programmer\_id` ASC) ,

INDEX `fk\_tb\_room\_member\_tb\_chatroom1\_idx` (`tb\_chatroom\_id` ASC) ,

CONSTRAINT `fk\_tb\_room\_member\_tb\_software\_programmer1`

FOREIGN KEY (`tb\_software\_programmer\_id`)

REFERENCES `programmer\_job`.`tb\_software\_programmer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_room\_member\_tb\_chatroom1`

FOREIGN KEY (`tb\_chatroom\_id`)

REFERENCES `programmer\_job`.`tb\_chatroom` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_message`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_message` (

`id` INT NOT NULL AUTO\_INCREMENT,

`text` LONGTEXT NOT NULL,

`created\_message` DATETIME NOT NULL,

`tb\_chatroom\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_message\_tb\_chatroom1\_idx` (`tb\_chatroom\_id` ASC) ,

CONSTRAINT `fk\_tb\_message\_tb\_chatroom1`

FOREIGN KEY (`tb\_chatroom\_id`)

REFERENCES `programmer\_job`.`tb\_chatroom` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_thread`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_thread` (

`id` INT NOT NULL AUTO\_INCREMENT,

`text` LONGTEXT NOT NULL,

`hastag` VARCHAR(45) NULL,

`created\_thread` DATETIME NOT NULL,

`tb\_software\_programmer\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_thread\_tb\_software\_programmer1\_idx` (`tb\_software\_programmer\_id` ASC) ,

CONSTRAINT `fk\_tb\_thread\_tb\_software\_programmer1`

FOREIGN KEY (`tb\_software\_programmer\_id`)

REFERENCES `programmer\_job`.`tb\_software\_programmer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_thread\_comment`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_thread\_comment` (

`id` INT NOT NULL AUTO\_INCREMENT,

`text` LONGTEXT NOT NULL,

`created\_comment` DATETIME NOT NULL,

`tb\_thread\_id` INT NOT NULL,

`tb\_software\_programmer\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_thread\_comment\_tb\_thread1\_idx` (`tb\_thread\_id` ASC) ,

INDEX `fk\_tb\_thread\_comment\_tb\_software\_programmer1\_idx` (`tb\_software\_programmer\_id` ASC) ,

CONSTRAINT `fk\_tb\_thread\_comment\_tb\_thread1`

FOREIGN KEY (`tb\_thread\_id`)

REFERENCES `programmer\_job`.`tb\_thread` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_thread\_comment\_tb\_software\_programmer1`

FOREIGN KEY (`tb\_software\_programmer\_id`)

REFERENCES `programmer\_job`.`tb\_software\_programmer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_job\_vacancy`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_job\_vacancy` (

`id` INT NOT NULL AUTO\_INCREMENT,

`title` VARCHAR(255) NOT NULL,

`description` LONGTEXT NOT NULL,

`category` VARCHAR(255) NOT NULL,

`status` VARCHAR(45) NOT NULL,

`created\_vacancy` DATETIME NOT NULL,

`tb\_software\_programmer\_id` INT NOT NULL,

`tb\_company\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_job\_vacancy\_tb\_software\_programmer1\_idx` (`tb\_software\_programmer\_id` ASC) ,

INDEX `fk\_tb\_job\_vacancy\_tb\_company1\_idx` (`tb\_company\_id` ASC) ,

CONSTRAINT `fk\_tb\_job\_vacancy\_tb\_software\_programmer1`

FOREIGN KEY (`tb\_software\_programmer\_id`)

REFERENCES `programmer\_job`.`tb\_software\_programmer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_job\_vacancy\_tb\_company1`

FOREIGN KEY (`tb\_company\_id`)

REFERENCES `programmer\_job`.`tb\_company` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

-- -----------------------------------------------------

-- Table `programmer\_job`.`tb\_job\_application`

-- -----------------------------------------------------

CREATE TABLE IF NOT EXISTS `programmer\_job`.`tb\_job\_application` (

`id` INT NOT NULL AUTO\_INCREMENT,

`files` VARCHAR(255) NULL,

`status` VARCHAR(45) NOT NULL,

`created\_application` DATETIME NOT NULL,

`tb\_job\_vacancy\_id` INT NOT NULL,

`tb\_software\_programmer\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `fk\_tb\_job\_vacancy\_tb\_job\_vacancy1\_idx` (`tb\_job\_vacancy\_id` ASC) ,

INDEX `fk\_tb\_job\_vacancy\_tb\_software\_programmer2\_idx` (`tb\_software\_programmer\_id` ASC) ,

CONSTRAINT `fk\_tb\_job\_vacancy\_tb\_job\_vacancy1`

FOREIGN KEY (`tb\_job\_vacancy\_id`)

REFERENCES `programmer\_job`.`tb\_job\_vacancy` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION,

CONSTRAINT `fk\_tb\_job\_vacancy\_tb\_software\_programmer2`

FOREIGN KEY (`tb\_software\_programmer\_id`)

REFERENCES `programmer\_job`.`tb\_software\_programmer` (`id`)

ON DELETE NO ACTION

ON UPDATE NO ACTION)

ENGINE = InnoDB;

SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

**Function**

Pada tugas kali ini saya membuat dua buah function pada sql, yaitu function menghitung jumlah thread yang telah dibuat oleh satu user, dan menghitung jumlah chat pada suatu grup. Berikut sourcenya :

**Menghitung jumlah thread :**

DELIMITER $$

CREATE FUNCTION countMyThread(

userId INT

)

RETURNS INT

DETERMINISTIC

BEGIN

return (Select count(id) FROM tb\_thread WHERE tb\_software\_programmer\_id = userId);

END $$

DELIMITER ;

**Menghitung jumlah pesan dalam satu grup :**

DELIMITER $$

CREATE FUNCTION countMyChat(

roomId INT

)

RETURNS INT

DETERMINISTIC

BEGIN

return (Select count(id) FROM tb\_message WHERE tb\_chatroom\_id = roomId);

END $$

DELIMITER ;

**Store Procedure**

Pada tugas kali ini saya juga membuat dua buah store procedure, yaitu unsend message, dan show all message. Store procedure unsend message digunakan untuk user menarik Kembali pesan yang telah dikirimkan. Pada procedure ini, isi pesan akan dirubah menjadi “Message was deleted”. Sedangkan pada procedure show all message digunakan untuk mengambil semua data chat pada grup tertentun. Berikut sourcenya :

**Unsent Message :**

DELIMITER $$

DROP PROCEDURE IF EXISTS `unsendMessage`$$

CREATE PROCEDURE unsendMessage(

IN messageId INT

)

BEGIN

UPDATE tb\_message SET text='Message was deleted' WHERE id=messageId;

END$$

DELIMITER ;

**Show All Message**

DELIMITER $$

DROP PROCEDURE IF EXISTS `showMyMessage`$$

CREATE PROCEDURE showMyMessage(

IN roomId INT

)

BEGIN

SELECT \* FROM tb\_message WHERE tb\_chatroom\_id=roomId;

END$$

DELIMITER ;

**Trigger**

Pada tugas kali ini saya membuat dua jenis trigger, yang pertama yaitu trigger yang berjalan saat perusahaan menghapus lowongan pekerjaan yang telah di post, tigger ini mengubah id job vacancy yang tersimpan pada tabel tb\_job\_application menjadi bernilai 0, dimana data lowongan yang memiliki id 0 berisikan “Sorry, this job has been deleted by company”. Sehingga tampilan pada user yang telah melamar pekerjaan pada lowongan tersebut menjadi “Sorry, this job has been deleted by company”.

Untuk trigger yang kedua yaitu trigger yang berjalan saat menambahkan data baru pada tabel tb\_admin, tb\_software\_programmer, tb\_job\_vacancy, tb\_job\_application, tb\_thread, tb\_message dan tb\_thread\_comment. Trigger ini akan otomatis mengisikan kolom “date” pada tabel-tabel tersebut, sehingga developer hanya cukup mengisi kolom yang lain.

Berikut adalah sourcenya:

**Trigger pada saat penghapusan lowongan**

DELIMITER $$

DROP TRIGGER IF EXISTS `setJVDelete`$$

CREATE TRIGGER setJADelete

BEFORE DELETE ON tb\_job\_vacancy FOR EACH ROW

BEGIN

UPDATE tb\_job\_application SET tb\_job\_vacancy\_id = '0' WHERE tb\_job\_vacancy\_id = OLD.id;

END$$

DELIMITER ;

**Trigger saat penambahan data baru :**

-- Trigger for update created account when insert new data in tb\_admin

DELIMITER $$

DROP TRIGGER IF EXISTS `setAdminCreatedDate`$$

CREATE TRIGGER setAdminCreatedDate

BEFORE INSERT ON tb\_admin FOR EACH ROW

BEGIN

SET NEW.created\_account = NOW();

END$$

DELIMITER ;

-- ----------------------------------------------------------

-- Trigger for update created account when insert new data in tb\_software\_programmer

DELIMITER $$

DROP TRIGGER IF EXISTS `setSPCreatedDate`$$

CREATE TRIGGER setSPCreatedDate

BEFORE INSERT ON tb\_software\_programmer FOR EACH ROW

BEGIN

SET NEW.created\_account = NOW();

END$$

DELIMITER ;

-- ----------------------------------------------------------

-- Trigger for update created account when insert new data in tb\_job\_vacancy

DELIMITER $$

DROP TRIGGER IF EXISTS `setJVCreatedDate`$$

CREATE TRIGGER setJVCreatedDate

BEFORE INSERT ON tb\_job\_vacancy FOR EACH ROW

BEGIN

SET NEW.created\_vacancy = NOW();

END$$

DELIMITER ;

-- ----------------------------------------------------------

-- Trigger for update created account when insert new data in tb\_job\_application

DELIMITER $$

DROP TRIGGER IF EXISTS `setJACreatedDate`$$

CREATE TRIGGER setJACreatedDate

BEFORE INSERT ON tb\_job\_application FOR EACH ROW

BEGIN

SET NEW.created\_application = NOW();

END$$

DELIMITER ;

-- ----------------------------------------------------------

-- Trigger for update created account when insert new data in tb\_thread

DELIMITER $$

DROP TRIGGER IF EXISTS `setThreadCreatedDate`$$

CREATE TRIGGER setThreadCreatedDate

BEFORE INSERT ON tb\_thread FOR EACH ROW

BEGIN

SET NEW.created\_thread = NOW();

END$$

DELIMITER ;

-- ----------------------------------------------------------

-- Trigger for update created account when insert new data in tb\_thread\_comment

DELIMITER $$

DROP TRIGGER IF EXISTS `setTCCreatedDate`$$

CREATE TRIGGER setTCCreatedDate

BEFORE INSERT ON tb\_thread\_comment FOR EACH ROW

BEGIN

SET NEW.created\_comment = NOW();

END$$

DELIMITER ;

-- ----------------------------------------------------------