Database Project

G24

ER Diagram

Please access the ER diagram from here

Indexes

Currently there is additional indexes except the primary keys

- Index for the username
- Index for the flight arrival time
- Index for the flight arrival time

```
CREATE INDEX idx_username ON registered_user (username);
CREATE INDEX idx_departure_time ON flight (departure_time);
CREATE INDEX idx_arrival_time ON flight (arrival_time);
```

Triggers

```
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER `count_bookings_trigger` BEFORE INSERT ON `booking` FOR EACH ROW BEGIN
    DECLARE user_bookings_count INT;
    DECLARE gold_threshold INT;
    DECLARE freq_threshold INT;
    DECLARE user_reg_id INT;

    DECLARE passenger_class VARCHAR(1);
    DECLARE tot_cost decimal(8, 2);
    DECLARE discount decimal(4, 2);


    /* First we make sure the user is a registered_user and not an unregistered customer */
    IF EXISTS (SELECT 1 FROM registered_user WHERE registered_user.customer_id = NEW.customer_id) THEN

        SET user_reg_id = (SELECT user_id FROM registered_user WHERE registered_user.customer_id = NEW.customer_id);

        IF NOT EXISTS (SELECT 1 FROM user_booking_count WHERE reg_user_id = user_reg_id) THEN
            INSERT INTO user_booking_count (reg_user_id, num_bookings)
            VALUES (user_reg_id, 0);
        ELSE
            UPDATE user_booking_count
            SET num_bookings = num_bookings + 1
            WHERE user_booking_count.reg_user_id = user_reg_id;
        END IF;

        /* Checking whether user is eligible for upgrade to 'Gold' or 'Frequent' membership */
        SET gold_threshold = (SELECT mem_type_threshold FROM membership WHERE mem_type = 'G');
        SET freq_threshold = (SELECT mem_type_threshold FROM membership WHERE mem_type = 'F');
        SET user_bookings_count = (SELECT num_bookings FROM user_booking_count WHERE user_booking_count.reg_user_id = user_reg_id);
```

```
        IF (user_bookings_count >= gold_threshold) THEN
            UPDATE registered_user
            SET membership = 'G'
            WHERE registered_user.user_id = user_reg_id;
        ELSE
            IF (user_bookings_count >= freq_threshold) THEN
                UPDATE registered_user
                SET membership = 'F'
                WHERE registered_user.user_id = user_reg_id;
            END IF;
        END IF;

    END IF;

    /* Calculating the cost of the booking */
    /* Set the 'discount' and 'passenger_class' values for our user. If user is a guest, discount=0 */
    IF EXISTS (SELECT 1 FROM registered_user WHERE registered_user.customer_id = NEW.customer_id) THEN
        /* Since user is registered, we set discount considering user's membership type */
        SET discount = (SELECT discount_rate FROM membership WHERE mem_type_id = (SELECT membership FROM registered_user WHERE registered_user.customer_id = NEW.customer_id));
    ELSE
        /* Means user is not registered, and therefore not eligible for discount */
        SET discount = 0;
    END IF;
    SET passenger_class = (SELECT seat_class FROM seat WHERE seat_id = NEW.seat_id);

    IF passenger_class = 'E' THEN
        SET tot_cost = (SELECT economy_price FROM route WHERE
                        route.route_id = (SELECT route_id FROM flight WHERE flight.flight_id = NEW.flight_id)) * (100 - discount) / 100;
    END IF;
```

```
    IF passenger_class = 'E' THEN
        SET tot_cost = (SELECT economy_price FROM route WHERE
                        route.route_id = (SELECT route_id FROM flight WHERE flight.flight_id = NEW.flight_id)) * (100 - discount) / 100;
    END IF;

    IF passenger_class = 'B' THEN
        SET tot_cost = (SELECT business_price FROM route WHERE
                        route.route_id = (SELECT route_id FROM flight WHERE flight.flight_id = NEW.flight_id)) * (100 - discount) / 100;
    END IF;

    IF passenger_class = 'P' THEN
        SET tot_cost = (SELECT platinum_price FROM route WHERE
                        route.route_id = (SELECT route_id FROM flight WHERE flight.flight_id = NEW.flight_id)) * (100 - discount) / 100;
    END IF;

    SET NEW.total_cost = tot_cost;

END */;;
DELIMITER ;
/*!50003 SET sql_mode              = @saved_sql_mode */ ;
/*!50003 SET character_set_client  = @saved_cs_client */ ;
/*!50003 SET character_set_results = @saved_cs_results */ ;
/*!50003 SET collation_connection  = @saved_col_connection */ ;
```

```
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER `seat_entry_trigger` AFTER INSERT ON `flight` FOR EACH ROW BEGIN
    DECLARE economy_seat_count INT;
    DECLARE business_seat_count INT;
    DECLARE platinum_seat_count INT;

    SET economy_seat_count = 1;
    SET business_seat_count = 1;
    SET platinum_seat_count = 1;

    WHILE economy_seat_count <= (SELECT economy_seats
                                 FROM aircraft_model AS m
                                 JOIN aircraft AS a ON m.model_id = a.model_id
                                 WHERE a.aircraft_id = new.aircraft_id)
    DO
        INSERT INTO seat(flight_id, seat_number, seat_class)
        VALUES (new.flight_id, economy_seat_count, 'E');
        set economy_seat_count = economy_seat_count + 1;
    end while;


    WHILE business_seat_count <= (SELECT business_seats
                                 FROM aircraft_model AS m
                                 JOIN aircraft AS a ON m.model_id = a.model_id
                                 WHERE a.aircraft_id = new.aircraft_id)
    DO
        INSERT INTO seat(flight_id, seat_number, seat_class)
        VALUES (new.flight_id, business_seat_count, 'B');
        set business_seat_count = business_seat_count + 1;
    end while;
```

```
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER `count_bookings_trigger` BEFORE INSERT ON `booking` FOR EACH ROW BEGIN
    DECLARE user_bookings_count INT;
    DECLARE gold_threshold INT;
    DECLARE freq_threshold INT;
    DECLARE user_reg_id INT;

    DECLARE passenger_class VARCHAR(1);
    DECLARE tot_cost decimal(8, 2);
    DECLARE discount decimal(4, 2);


    /* First we make sure the user is a registered_user and not an unregistered customer */
    IF EXISTS (SELECT 1 FROM registered_user WHERE registered_user.customer_id = NEW.customer_id) THEN

        SET user_reg_id = (SELECT user_id FROM registered_user WHERE registered_user.customer_id = NEW.customer_id);

        IF NOT EXISTS (SELECT 1 FROM user_booking_count WHERE reg_user_id = user_reg_id) THEN
            INSERT INTO user_booking_count (reg_user_id, num_bookings)
            VALUES (user_reg_id, 0);
        ELSE
            UPDATE user_booking_count
            SET num_bookings = num_bookings + 1
            WHERE user_booking_count.reg_user_id = user_reg_id;
        END IF;

        /* Checking whether user is eligible for upgrade to 'Gold' or 'Frequent' membership */
        SET gold_threshold = (SELECT mem_type_threshold FROM membership WHERE mem_type = 'G');
        SET freq_threshold = (SELECT mem_type_threshold FROM membership WHERE mem_type = 'F');
        SET user_bookings_count = (SELECT num_bookings FROM user_booking_count WHERE user_booking_count.reg_user_id = user_reg_id);
```

```
    WHILE platinum_seat_count <= (SELECT platinum_seats
                                  FROM aircraft_model AS m
                                  JOIN aircraft AS a ON m.model_id = a.model_id
                                  WHERE a.aircraft_id = new.aircraft_id)
    DO
        INSERT INTO seat(flight_id, seat_number, seat_class)
        VALUES (new.flight_id, platinum_seat_count, 'P');
        set platinum_seat_count = platinum_seat_count + 1;
    end while;

end */;;
DELIMITER ;
/*!50003 SET sql_mode              = @saved_sql_mode */ ;
/*!50003 SET character_set_client  = @saved_cs_client */ ;
/*!50003 SET character_set_results = @saved_cs_results */ ;
/*!50003 SET collation_connection  = @saved_col_connection */ ;
```

```
/*!50003 CREATE*/ /*!50017 DEFINER=`root`@`localhost`*/ /*!50003 TRIGGER `track_bookings_count` AFTER INSERT ON `registered_user` FOR EACH ROW BEGIN
    IF NOT EXISTS (SELECT 1 FROM user_booking_count WHERE reg_user_id = NEW.user_id) THEN
        INSERT INTO user_booking_count (reg_user_id, num_bookings)
        VALUES (NEW.user_id, 0);
    END IF;
END */;;
DELIMITER ;
/*!50003 SET sql_mode              = @saved_sql_mode */ ;
/*!50003 SET character_set_client  = @saved_cs_client */ ;
/*!50003 SET character_set_results = @saved_cs_results */ ;
/*!50003 SET collation_connection  = @saved_col_connection */ ;
```

Procedures

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `CreateBooking`(IN in_customer_id INT(11), IN in_flight_id INT(11), IN in_seat_id INT(11), IN in_payment_status BIT(1), OUT out_booking_id INT(11))
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        @sqlstate = RETURNED_SQLSTATE, @errno = MYSQL_ERRNO, @text = MESSAGE_TEXT;
        SELECT @sqlstate, @errno, @text;
        ROLLBACK;
    END;

    START TRANSACTION;
        -- Create a booking
        INSERT INTO booking (customer_id, flight_id, seat_id, payment_status) VALUES (in_customer_id, in_flight_id, in_seat_id, in_payment_status);
        SET out_booking_id = LAST_INSERT_ID();

        -- Update the availability of the seat
UPDATE seat
SET
    `availability` = 0
WHERE
    seat_id = in_seat_id;
    COMMIT;

END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `CreateUserPassenger`(IN in_user_id INT(11), IN in_user_type VARCHAR(10), IN in_name VARCHAR(64), IN in_dob DATE, IN in_address
                                                    IN in_nic VARCHAR(20), IN in_passport_id VARCHAR(20), OUT out_customer_id INT(11))
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        @sqlstate = RETURNED_SQLSTATE, @errno = MYSQL_ERRNO, @text = MESSAGE_TEXT;
        SELECT @sqlstate, @errno, @text;
        ROLLBACK;
    END;

    -- First we insert into customer table and then insert into user_passenger table
    START TRANSACTION;
        INSERT INTO customer (user_type, name, date_of_birth, address, nic, passport_id) VALUES (in_user_type, in_name, in_dob, in_address, in_nic, in_passport_id);
        SET out_customer_id = LAST_INSERT_ID();

        INSERT INTO user_passenger (customer_id, registered_user_id) VALUES (out_customer_id, in_user_id);

    COMMIT;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertAndGetGuestID`(IN in_user_type VARCHAR(10), IN in_name VARCHAR(64), IN in_dob DATE, IN in_address VARCHAR(128), I
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        @sqlstate = RETURNED_SQLSTATE, @errno = MYSQL_ERRNO, @text = MESSAGE_TEXT;
        SELECT @sqlstate, @errno, @text;
        ROLLBACK;
    END;

    START TRANSACTION;
        INSERT INTO customer (user_type, name, date_of_birth, address, nic, passport_id) VALUES (in_user_type, in_name, in_dob, in_address, in_nic, in_passport_id);
        SET out_customer_id = LAST_INSERT_ID();
    COMMIT;
END
```

```
CREATE DEFINER=`root`@`localhost` PROCEDURE `GetHashedPass`(IN in_username VARCHAR(45), OUT hashed_pass VARCHAR(72))
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        GET DIAGNOSTICS CONDITION 1
        @sqlstate = RETURNED_SQLSTATE, @errno = MYSQL_ERRNO, @text = MESSAGE_TEXT;
        SELECT @sqlstate, @errno, @text;
    END;

    SELECT password INTO hashed_pass FROM registered_user WHERE registered_user.username = in_username LIMIT 1;

    IF hashed_pass IS NULL THEN
        SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'User not found';
    END IF;
END
```

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `LoginUser`(IN in_username VARCHAR(45))
BEGIN
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;
    END;

    -- insert into something
    -- update something else

    START TRANSACTION;
        UPDATE registered_user SET login_status = 1 WHERE registered_user.username = in_username;
    COMMIT;

    SELECT user_id, username, membership, customer_id FROM registered_user WHERE registered_user.username = in_username;
END
```

```sql
CREATE DEFINER=`root`@`localhost` PROCEDURE `InsertNewUser`(IN user_type VARCHAR(10), IN name VARCHAR(64), IN dob DATE, IN address VARCHAR(128), IN nic VARCHAR(20), IN passport_id VARCHAR(20),
                                    IN membership VARCHAR(1), IN username VARCHAR(45), IN password VARCHAR(72))
BEGIN
    DECLARE customer_id INT DEFAULT 0;

    -- Declare an error handler to rollback in case of critical failure
    DECLARE CONTINUE HANDLER FOR SQLEXCEPTION
    BEGIN
        ROLLBACK;

    END;

    -- Start a transaction
    START TRANSACTION;

        -- Insert into customer table
        INSERT INTO customer (user_type, name, date_of_birth, address, nic, passport_id)
        VALUES (user_type, name, dob, address, nic, passport_id);

        -- Get the last inserted customer_id
        SET customer_id = LAST_INSERT_ID();

        -- Insert into registered_user table
        INSERT INTO registered_user (membership, username, password, login_status, customer_id)
        VALUES (membership, username, password, 0, customer_id);

    -- Commit the transaction
    COMMIT;
END
```

Views

```
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `user_view` AS
    SELECT
        `registered_user`.`user_id` AS `user_id`,
        `registered_user`.`membership` AS `membership`,
        `registered_user`.`username` AS `username`,
        `registered_user`.`login_status` AS `login_status`,
        `registered_user`.`customer_id` AS `customer_id`
    FROM
        `registered_user`
```