LAPORAN

DATABASE

UNTUK MEMENUHI UAS

Mata kuliah : Basis Data

Dosen Pengampu : Mohamad Firdaus, M.Kom

Penyusun

Dimas Dwi Rianto (623C0004)

INSTITUT TEKNOLOGI DAN KESEHATAN MAHARDIKA

PROGRAM STUDI INFORMATIKA S1

2024

1. Rancangan Struktur Database

Perancangan struktur database kali ini saya lakukan dimulai dengan memberi nama database saya sebagai “deltaromeo1”, database ini saya buat guna mendukung dan memenuhi kebutuhan aplikasi web atau aplikasi desktop (dengan python). Dalam perancangan database ini juga saya melakukan Normalisasi, Pembuatan ERD dan Pembuatan Struktur Tabel menggunakan SQL. Berikut adalah hasil normalisasi tabel dari 1NF sampai dengan 3NF

Tabel Sebelum Normalisasi

transaction_id	user_id	id_barang	id_guide	total_harga	status	created_at
DTR001	US001	BRG001 BRG002	GD001	Rp5.000.000	Completed	01/01/2024 12:00
DTR002	US002	BRG003	GD002 GD003	Rp2.500.000	Pending	02/01/2024 13:00

Berikut adalah tabel yang belum dinormalisasi untuk selanjutnya dilakukan normalisasi tabel mulai dari 1NF sampai dengan 3NF

Tabel 1NF(transactions)

transaction_id	user_id	item_type	item_id	total_harga	status	created_at
DTR001	US001	BARANG	BRG001	Rp5.000.000	Completed	01/01/2024 12:00
DTR001	US001	BARANG	BRG002	Rp5.000.000	Completed	01/01/2024 12:00
DTR001	US001	GUIDE	GD001	Rp5.000.000	Completed	01/01/2024 12:00
DTR002	US002	BARANG	BRG003	Rp2.500.000	Pending	02/01/2024 13:00
DTR002	US002	GUIDE	GD002	Rp2.500.000	Pending	02/01/2024 13:00
DTR002	US002	GUIDE	GD003	Rp2.500.000	Pending	02/01/2024 13:00

Tabel di atas merupakan hasil Normalisasi 1NF, Normalisasi 1NF dilakukan untuk memisahkan alat camping dan jasa guide dalam baris terpisah, setiap kolom hanya memiliki satu nilai per sel.

Tabel 2NF

tbl_transactions

transaction_id	user_id	total_harga	status	created_at
DTR001	US001	Rp5.000.000	Completed	01/01/2024 12:00
DTR002	US002	Rp2.500.000	Pending	02/01/2024 13:00

tbl transaction_items

transactions_id	item_type	item_id
DTR001	BARANG	BRG001
DTR001	BARANG	BRG002
DTR001	GUIDE	GD001
DTR002	BARANG	BRG003
DTR002	GUIDE	GD002
DTR002	GUIDE	GD003

Pada tahap ini kita memastikan bahwa setiap atribut non-primer sepenuhnya bergantung pada primary key. Berdasarkan tabel yang sudah ada pada 1NF, kita pisahkan menjadi dua tabel utama.

Untuk selanjutnya kita ke normalisasi 3NF, di tabel transactions kita memiliki kolom status yang berisi informasi tentang status transaksi. Ini menyebabkan ketergantungan transitif, karena status dalam tabel transactions bisa tergantung pada atribut non-primer lainnya, seperti status_name yang seharusnya ada dalam tabel terpisah.

Perbaikan untuk di 3NF Untuk memenuhi 3NF, kita perlu memecah informasi yang berhubungan dengan status transaksi ke dalam tabel yang terpisah. Tabel transaction_statuses akan menyimpan status dan status_id, yang kemudian akan digunakan sebagai foreign key di tabel transactions.

Tabel 3NF

tbl_transactions

transaction_id	user_id	total_harga	status_id	created_at
DTR001	US001	Rp5.000.000	1	01/01/2024 12:00
DTR002	US002	Rp2.500.000	2	02/01/2024 13:00

tbl_transactions_status

status_id	status_name
1	Completed
2	Pending
3	Cancelled

tbl_transactions_items

transaction_id	item_type	item_id
DTR001	BARANG	BRG001
DTR001	BARANG	BRG002
DTR001	GUIDE	GD001
DTR002	BARANG	BRG003
DTR002	GUIDE	GD002
DTR002	GUIDE	GD003

Dengan perubahan ini, kita memastikan bahwa tidak ada ketergantungan transitif. Dalam tabel transactions, status_id sekarang langsung merujuk ke tabel transaction_statuses, yang menjelaskan status transaksi dengan menggunakan foreign key status_id. Sebelumnya (2NF), kita memiliki kolom status dalam tabel transactions, yang bisa tergantung pada informasi lain, dan ini merupakan ketergantungan transitif. Setelah (3NF), kita memisahkan informasi status ke tabel terpisah (transaction_statuses), yang menjamin bahwa setiap kolom dalam tabel transactions hanya bergantung langsung pada primary key transaction_id.

2. Query SQL Untuk Memproses Data Kasus Bisnis

Saya menggunakan 2 Query SQL yaitu JOIN dan Subquery untuk memproses data pada kasus bisnis saya adapun query yang saya buat dengan JOIN adalah sebagai berikut :

2.1 JOIN

1. Menampilkan daftar transaksi dan detail barang yang dibeli oleh pengguna

```sql
SELECT
    u.username AS nama_pengguna,
    t.transaction_id AS id_transaksi,
    i.nama_barang AS nama_barang,
    ti.quantity AS jumlah_barang,
    ti.subtotal AS total_harga_per_item
FROM
    users u
JOIN
    transactions t ON u.user_id = t.user_id
JOIN
    transactions_items ti ON t.transaction_id = ti.transaction_id
JOIN
    barang i ON ti.id_barang = i.id_barang
ORDER BY
    t.created_at DESC;
```

- Tabel yang Digunakan : users, transactions, transaction_items, dan items.
- JOIN digunakan untuk menggabungkan tabel users, transactions, transaction_items, dan items untuk mendapatkan informasi lengkap tentang transaksi dan barang yang dibeli.
- Hasil: Menampilkan nama pengguna, ID transaksi, nama barang, jumlah barang, dan subtotal untuk setiap item.

2. Menampilkan Total Pengeluaran Setiap Pengguna

```sql
SELECT
    u.username AS nama_pengguna,
    SUM(t.total_harga) AS total_pengeluaran
FROM
    users u
JOIN
    transactions t ON u.user_id = t.user_id
GROUP BY
    u.username
ORDER BY
    total_pengeluaran DESC;
```

- Tabel yang Digunakan: users dan transactions.
- JOIN menghubungkan tabel users dengan tabel transactions untuk mendapatkan total pengeluaran dari setiap pengguna.
- GROUP BY digunakan untuk mengelompokkan data berdasarkan pengguna.

- Hasil: Menampilkan nama pengguna dan total pengeluaran yang diurutkan dari yang terbesar ke terkecil.

3. Menampilkan status transaksi beserta jumlah nya

```sql
SELECT
    ts.status_name AS status_transaksi,
    COUNT(t.transaction_id) AS jumlah_transaksi
FROM
    transactions t
JOIN
    transaction_status ts ON t.status_order = ts.status_id
GROUP BY
    ts.status_name
ORDER BY
    jumlah_transaksi DESC;
```

- Tabel yang Digunakan: transactions dan transaction_status.
- JOIN menghubungkan tabel transactions dengan tabel transaction_status untuk mendapatkan nama status transaksi.
- GROUP BY digunakan untuk mengelompokkan data berdasarkan status transaksi.
- Hasil: Menampilkan jumlah transaksi untuk setiap status transaksi.

4. Menampilkan Transaksi Terakhir yang dilakukan oleh setiap pengguna

```sql
SELECT
    u.username AS nama_pengguna,
    t.transaction_id AS id_transaksi,
    i.nama_barang AS nama_barang,
    ti.quantity AS jumlah_barang,
    ti.subtotal AS total_harga_per_item
FROM
    users u
JOIN
    transactions t ON u.user_id = t.user_id
JOIN
    transactions_items ti ON t.transaction_id = ti.transaction_id
JOIN
    barang i ON ti.id_barang = i.id_barang
ORDER BY
    t.created_at DESC;
```

- Tabel yang Digunakan: transactions dan users.
- JOIN menghubungkan tabel transactions dengan tabel users untuk mendapatkan informasi pengguna.
- Subquery digunakan untuk mendapatkan waktu transaksi terakhir (MAX(created_at)) untuk setiap pengguna.
- Hasil: Menampilkan ID transaksi terakhir, nama pengguna, total transaksi, dan waktu transaksi.

5. Menampilan barang dengan penjualan tertinggi

```sql
SELECT
    i.nama_barang AS nama_barang,
    SUM(ti.quantity) AS total_terjual,
    i.harga AS harga_per_unit
FROM
    transactions_items ti
JOIN
    barang i ON ti.id_barang = i.id_barang
GROUP BY
    i.nama_barang, i.harga
ORDER BY
    total_terjual DESC
LIMIT 5;
```

- Tabel yang Digunakan: transaction_items dan items.
- JOIN menghubungkan tabel transaction_items dengan tabel items untuk mendapatkan nama barang dan harga per unit.
- GROUP BY digunakan untuk mengelompokkan data berdasarkan nama barang.
- Hasil: Menampilkan nama barang, total barang yang terjual, dan harga per unit, diurutkan dari yang paling laku.

2.2 Subquery

1. Subquery Pada SELECT

```sql
SELECT
    t.transaction_id AS id_transaksi,
    t.total_harga AS total_harga,
    t.created_at AS tanggal_transaksi,
    (
        SELECT SUM(quantity)
        FROM transactions_items
        WHERE transactions_items.transaction_id = t.transaction_id
    ) AS jumlah_barang
FROM
    transactions t;
```

- Subquery pada bagian SELECT menghitung jumlah barang (SUM(quantity)) dari tabel transaction_items berdasarkan transaction_id pada tabel transactions.
- Hasilnya akan menunjukkan data transaksi beserta jumlah barang yang dibeli.

2. Subquery pada WHERE

```
SELECT
    t.transaction_id AS id_transaksi,
    t.total_harga AS total_harga,
    t.created_at AS tanggal_transaksi
FROM
    transactions t
WHERE
    t.total_harga > (
        SELECT AVG(total_harga)
        FROM transactions
    );
```

- Subquery pada bagian WHERE menghitung rata-rata total harga (AVG(total_harga)) dari tabel transactions.
- Query utama hanya akan menampilkan transaksi yang total harganya lebih besar dari rata-rata tersebut.

3. Subquery pada FROM

```
SELECT
    u.username AS nama_pengguna,
    t.jumlah_transaksi
FROM
    users u
JOIN
    (
        SELECT
            user_id,
            COUNT(transaction_id) AS jumlah_transaksi
        FROM
            transactions
        GROUP BY
            user_id
    ) t ON u.user_id = t.user_id;
```

- Subquery pada bagian FROM menghitung jumlah transaksi (COUNT(transaction_id)) untuk setiap pengguna (user_id).
- Query utama kemudian menggabungkan data dari tabel users untuk menampilkan nama pengguna beserta jumlah transaksi mereka.

3. Implementasi Database Object

Berikut adalah implementasi untuk memenuhi minimal tiga jenis database object pada database yang Anda gunakan. Saya akan memberikan contoh penggunaan View, Procedure, Function, Trigger, dan Event.

1. View

   Membuat view untuk menampilkan daftar transaksi beserta nama pengguna dan total harga transaksi

```sql
CREATE VIEW view_user_transactions AS
SELECT
    u.username AS nama_pengguna,
    t.transaction_id AS id_transaksi,
    t.total_harga AS total_harga_transaksi,
    t.created_at AS waktu_transaksi
FROM
    users u
JOIN
    transactions t ON u.user_id = t.user_id;
```

   View ini akan menampilkan data transaksi beserta nama pengguna, total harga, dan waktu transaksi tanpa perlu menuliskan JOIN Query berulang kali.

2. Procedure

   Membuat procedure untuk menambahkan data transaksi baru beserta item yang dibeli dalam satu proses.

```sql
DELIMITER //
CREATE PROCEDURE add_transaction (
    IN p_user_id INT,
    IN p_total_harga DECIMAL(10,2),
    IN p_status_order VARCHAR(50),
    IN p_created_at DATETIME,
    IN p_item_id VARCHAR(50),
    IN p_quantity INT,
    IN p_subtotal DECIMAL(10,2)
)
BEGIN
    -- Tambahkan transaksi ke tabel transactions
    INSERT INTO transactions (user_id, total_harga, status_order, created_at)
    VALUES (p_user_id, p_total_harga, p_status_order, p_created_at);

    -- Ambil ID transaksi terakhir
    SET @transaction_id = LAST_INSERT_ID();

    -- Tambahkan item ke tabel transaction_items
    INSERT INTO transaction_items (transaction_id, item_id, quantity, subtotal)
    VALUES (@transaction_id, p_item_id, p_quantity, p_subtotal);
END //
DELIMITER ;
```

   Procedure ini secara otomatis akan menambahkan transaksi ke tabel transactions dan item terkait ke tabel transaction_items.

3. Function

Membuat function untuk menghitung total pengeluaran dari seorang pengguna berdasarkan user_id.

```
DELIMITER //
CREATE FUNCTION calculate_user_spending (p_user_id INT)
RETURNS DECIMAL(10,2)
DETERMINISTIC
BEGIN
    DECLARE total_spending DECIMAL(10,2);

    -- Hitung total pengeluaran
    SELECT SUM(total_harga) INTO total_spending
    FROM transactions
    WHERE user_id = p_user_id;

    RETURN IFNULL(total_spending, 0);
END //
DELIMITER ;
```

Function ini akan menghitung total pengeluaran dari pengguna dengan user_id tertentu.

4. Trigger

Membuat trigger untuk mengurangi stok barang secara otomatis setiap kali ada transaksi barang baru.

```
DELIMITER //
CREATE TRIGGER after_insert_transaction_item
AFTER INSERT ON transactions_items
FOR EACH ROW
BEGIN
    -- Kurangi stok barang
    UPDATE items
    SET stok = stok - NEW.quantity
    WHERE item_id = NEW.id_barang;
END //
DELIMITER ;
```

Trigger ini secara otomatis akan mengurangi stok barang di tabel items ketika ada data baru yang dimasukkan ke tabel transaction_items.

5. Event

```
DELIMITER //
CREATE EVENT delete_old_pending_transactions
ON SCHEDULE EVERY 1 DAY
DO
BEGIN
    DELETE FROM transactions
    WHERE status_order = 'Pending' AND created_at < (NOW() - INTERVAL 7 DAY);
END //
DELIMITER ;
```

Event ini akan berjalan setiap hari dan menghapus transaksi yang berstatus "Pending" lebih dari 7 hari secara otomatis.