LAPORAN UAS SEMESTER 1 SISTEM BASIS DATA TOKO BANGUNAN

UNIVERSITAS
DUTA BANGSA
SURAKARTA

Disusun oleh:

NAMA	**: Hafidz Fathul Izzi**
	: Rasendriya Giri Mursi
NIM	**: 250119010**
	: 250119021
KELAS	**: TRPL25A1**

PROGRAM STUDI TEKNOLOGI REKAYASA PERANGKAT LUNAK

FAKULTAS ILMU KOMPUTER

UNIVERSITAS DUTA BANGSA SURAKARTA

2025

DAFTAR ISI :

BAB 1 PENDAHULUAN

1.1 Latar Belakang

Perkembangan bisnis toko bangunan memerlukan sistem pencatatan transaksi yang terstruktur dan efisien. Basis data digunakan untuk mengelola data pelanggan, barang, dan transaksi penjualan agar dapat diakses, dimanipulasi, dan dilaporkan dengan mudah.

1.2 Tujuan Proyek

- Merancang dan mengimplementasikan basis data untuk toko bangunan.
- Menerapkan konsep relasi tabel, normalisasi, dan query SQL.
- Menghasilkan laporan transaksi dan analisis penjualan.

1.3 Ruang Lingkup dan Batasan

- Sistem hanya mencakup transaksi penjualan.
- Data yang dikelola: pelanggan, barang, transaksi, detail transaksi.
- Tidak mencakup pembelian stok, retur, atau inventaris real-time.

1.4 Gambaran Umum Sistem

Sistem ini terdiri dari empat tabel utama:

- pelanggan → menyimpan data pembeli.
- barang → menyimpan daftar produk.
- transaksi → menyimpan data transaksi penjualan.
- detail_transaksi → menyimpan detail barang yang dibeli.

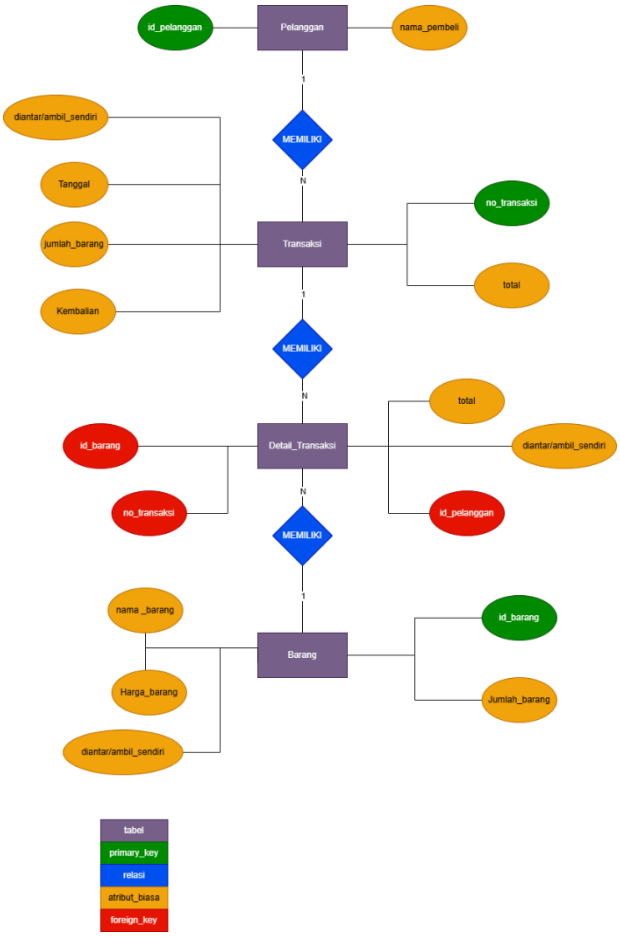
BAB 2 LANDASAN TEORI

2.1 Konsep Basis Data dan DBMS

Basis data adalah kumpulan data yang terorganisir yang dapat diakses, dikelola, dan diperbarui. Database Management System (DBMS) adalah perangkat lunak yang digunakan untuk mengelola basis data. MySQL merupakan salah satu DBMS relasional open-source yang menggunakan Structured Query Language (SQL) untuk mengelola data.

2.2 ERD dan Relasi

Entity Relationship Diagram (ERD) adalah model konseptual yang menggambarkan hubungan antara entitas dalam suatu sistem. Dalam proyek ini, ERD digunakan untuk memodelkan hubungan antara pelanggan, barang, transaksi, dan detail transaksi.



Gambar 2.1: ERD Final Sistem Database Toko Bangunan

Keterangan ERD:

1. Entitas: Pelanggan, Transaksi, Barang, dan Detail_Transaksi.
2. Relasi:
    1. Pelanggan melakukan banyak Transaksi (1:N).
    2. Satu Transaksi terdiri dari banyak Detail_Transaksi (1:N):
    3. Satu Barang dapat dijual dalam banyak Detail_Transaksi (1:N).
3. Primary Key ditandai.warna hijau
4. Foreign Key menghubungkan satu tabel ke tabel lainnya.ditandai dengan warna merah

2.3 Normalisasi

Normalisasi adalah proses pengorganisasian data dalam basis data untuk mengurangi redundansi dan meningkatkan integritas data. Proyek ini menerapkan normalisasi hingga bentuk ketiga (3NF):

- Bentuk Pertama (1NF): Setiap tabel memiliki primary key dan semua atribut bernilai atomik
- Bentuk Kedua (2NF): Memenuhi 1NF dan semua atribut non-kunci bergantung sepenuhnya pada primary key
- Bentuk Ketiga (3NF): Memenuhi 2NF dan tidak ada ketergantungan transitif antar atribut non-kunci

2.4 SQL dan Koneksi Database

SQL (Structured Query Language) adalah bahasa standar untuk mengelola basis data relasional. Koneksi database merupakan mekanisme untuk menghubungkan aplikasi dengan DBMS. Dalam proyek ini digunakan:

- DDL (Data Definition Language) untuk mendefinisikan struktur database
- DML (Data Manipulation Language) untuk memanipulasi data
- TCL (Transaction Control Language) untuk mengontrol transaksi
- Query dengan fungsi agregasi, GROUP BY, dan HAVING

BAB 3 PERANCANGAN DAN IMPLEMENTASI

3.1 Kebutuhan Sistem

Sistem database toko bangunan dirancang untuk memenuhi kebutuhan berikut:

1. Fungsional:

- Penyimpanan data pelanggan dan barang
- Pencatatan transaksi penjualan
- Penyimpanan detail barang yang dibeli
- Generasi laporan penjualan

2.Non-Fungsional:

- Responsif dalam pemrosesan query
- Data konsisten dengan integritas referensial
- Skalabel untuk penambahan data

3.2 Perancangan Database

Database dirancang dengan 4 tabel utama yang telah dinormalisasi hingga 3NF:

- Tabel pelanggan: Menyimpan identitas pelanggan
- Tabel barang: Menyimpan informasi produk
- Tabel transaksi: Mencatat header transaksi
- Tabel detail_transaksi: Mencatat detail item yang dibeli

Primary Key (PK) dan Foreign Key (FK) yang diterapkan:

- PK pelanggan: id_pelanggan
- PK barang: id_barang
- PK transaksi: no_transaksi
- PK detail_transaksi: id_detail (auto increment)
- FK: id_pelanggan di tabel transaksi merujuk ke tabel pelanggan
- FK: no_transaksi di detail_transaksi merujuk ke tabel transaksi
- FK: id_barang di detail_transaksi merujuk ke tabel barang

3.3 Implementasi Database

3.3.1 DDL (Data Definition Language)

Struktur tabel diimplementasikan menggunakan perintah CREATE TABLE dengan constraint yang sesuai:

```
4  CREATE TABLE pelanggan (
5      id_pelanggan VARCHAR(10) PRIMARY KEY,
6      nama_pembeli VARCHAR(100) NOT NULL
7  );
8
```

Gambar 3.1: Struktur Tabel Pelanggan

```
9  CREATE TABLE barang (
10     id_barang VARCHAR(10) PRIMARY KEY,
11     nama_barang VARCHAR(100) NOT NULL,
12     harga DECIMAL(12,2) NOT NULL
13 );
```

Gambar 3.2: Struktur Tabel Barang

```
15 CREATE TABLE transaksi (
16     no_transaksi VARCHAR(20) PRIMARY KEY,
17     tanggal DATE NOT NULL,
18     id_pelanggan VARCHAR(10) NOT NULL,
19     jenis_pengiriman VARCHAR(20) NOT NULL,
20     FOREIGN KEY (id_pelanggan) REFERENCES pelanggan(id_pelanggan)
21 );
```

Gambar 3.3: Struktur Tabel Transaksi

```
23 CREATE TABLE detail_transaksi (
24     id_detail INT AUTO_INCREMENT PRIMARY KEY,
25     no_transaksi VARCHAR(20) NOT NULL,
26     id_barang VARCHAR(10) NOT NULL,
27     jumlah_barang INT NOT NULL,
28     FOREIGN KEY (no_transaksi) REFERENCES transaksi(no_transaksi),
29     FOREIGN KEY (id_barang) REFERENCES barang(id_barang)
30 );
```

Gambar 3.4: Struktur Tabel Detail Transaksi

Keterangan: Setiap tabel menunjukkan definisi kolom, tipe data, constraint NOT NULL, dan hubungan PK-FK yang telah diterapkan dengan benar.

3.3.2 Seed Data

Data contoh dimasukkan untuk pengujian sistem:

- 1 data pelanggan: PLG001 - Mas Agus
- 5 data barang dengan harga berbeda
- 5 transaksi dengan jenis pengiriman berbeda
- 5 detail transaksi dengan jumlah barang bervariasi

```sql
INSERT INTO pelanggan (id_pelanggan, nama_pembeli) VALUES
('PLG001', 'Mas Agus');

INSERT INTO barang (id_barang, nama_barang, harga) VALUES
('BRG001', 'Pasir', 208000),
('BRG002', 'Tali Tambang', 6000),
('BRG003', 'Ember Oranye', 9000),
('BRG004', 'Semen', 52000),
('BRG005', 'Solasi', 3500);

INSERT INTO transaksi (no_transaksi, tanggal, id_pelanggan, jenis_pengiriman) VALUES
('701', '2025-09-08', 'PLG001', 'diantar'),
('702', '2025-09-08', 'PLG001', 'ambil_sendiri'),
('703', '2025-09-08', 'PLG001', 'ambil_sendiri'),
('704', '2025-09-08', 'PLG001', 'diantar'),
('705', '2025-09-08', 'PLG001', 'ambil_sendiri');
```

Gambar 3.5: Data contoh dimasukkan untuk pengujian sistem

3.3.3 Query Penting

Beberapa query penting yang diimplementasikan:

1. Query JOIN 4 Tabel

Pengertian

JOIN digunakan untuk menggabungkan data dari dua atau lebih tabel yang saling berelasi berdasarkan kolom tertentu (biasanya primary key dan foreign key). JOIN 4 tabel berarti menggabungkan empat tabel sekaligus dalam satu query.

```sql
SELECT t.no_transaksi, t.tanggal, p.nama_pembeli,
       b.nama_barang, dt.jumlah_barang, b.harga,
       (dt.jumlah_barang * b.harga) AS total
FROM transaksi t
JOIN pelanggan p ON t.id_pelanggan = p.id_pelanggan
JOIN detail_transaksi dt ON t.no_transaksi = dt.no_transaksi
JOIN barang b ON dt.id_barang = b.id_barang
ORDER BY t.tanggal, t.no_transaksi;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

no_transaksi	tanggal	nama_pembeli	nama_barang	jumlah_barang	harga	total
701	2025-09-08	Mas Agus	Pasir	1	208000.00	208000.00
702	2025-09-08	Mas Agus	Tali Tambang	1	6000.00	6000.00
703	2025-09-08	Mas Agus	Ember Oranye	5	9000.00	45000.00
704	2025-09-08	Mas Agus	Semen	15	52000.00	780000.00
705	2025-09-08	Mas Agus	Solasi	2	3500.00	7000.00

Gambar 3.6: Query JOIN 4 Tabel dan hasil

Keterangan:

- JOIN pelanggan → menghubungkan transaksi dengan pelanggan

- JOIN detail_transaksi → menghubungkan transaksi dengan detail barang
- JOIN barang → mengambil informasi barang
- Hasil query menampilkan data gabungan dari 4 tabel

Hasil Query:

Hasil query menampilkan beberapa transaksi atas nama Mas Agus pada tanggal 8 September 2025, dengan rincian pembelian berbagai barang seperti pasir, tali tambang, ember oranye, semen, dan solasi beserta jumlah dan total harganya.

Query ini memperlihatkan bagaimana JOIN 4 tabel digunakan untuk menghasilkan data transaksi yang detail dan terstruktur.

2. Query Agregasi dengan GROUP BY dan HAVING

Pengertian

Query ini digunakan untuk mengelompokkan data (GROUP BY) sekaligus menyaring hasil pengelompokan (HAVING) berdasarkan hasil fungsi agregat.

```
232 •   SELECT t.id_pelanggan, p.nama_pembeli,
233             COUNT(DISTINCT t.no_transaksi) AS frekuensi_beli,
234             SUM(dt.jumlah_barang) AS total_barang,
235             SUM(dt.jumlah_barang * b.harga) AS total_belanja
236     FROM transaksi t
237     JOIN pelanggan p ON t.id_pelanggan = p.id_pelanggan
238     JOIN detail_transaksi dt ON t.no_transaksi = dt.no_transaksi
239     JOIN barang b ON dt.id_barang = b.id_barang
240     GROUP BY t.id_pelanggan, p.nama_pembeli
241     HAVING SUM(dt.jumlah_barang * b.harga) > 100000;
242
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

id_pelanggan	nama_pembeli	frekuensi_beli	total_barang	total_belanja
PLG001	Mas Agus	5	24	1046000.00

Gambar 3.7: Query Agregasi dengan GROUP BY dan HAVING serta hasil

Keterangan:

- SUM() → menghitung total belanja pelanggan
- GROUP BY pelanggan.nama_pembeli → mengelompokkan data berdasarkan pelanggan
- HAVING → memfilter hasil pengelompokan
- JOIN → menggabungkan data dari beberapa tabel
- Query hanya menampilkan pelanggan dengan total belanja di atas 1.000.000

Hasil Query:

Berdasarkan hasil yang ditampilkan, pelanggan Mas Agus dengan ID PLG001 memiliki:

- Frekuensi pembelian sebanyak 5 kali
- Total barang yang dibeli sebanyak 24
- Total belanja sebesar Rp 104.600,00

Query ini membuktikan bahwa HAVING digunakan untuk memfilter hasil setelah proses agregasi dilakukan.

3. Fungsi Agregat Lengkap

Pengertian

Fungsi agregat adalah fungsi dalam SQL yang digunakan untuk mengolah banyak baris data menjadi satu nilai ringkasan. Fungsi ini sering digunakan untuk analisis data, laporan, dan statistik.

```
244 •   SELECT COUNT(*) AS jumlah_transaksi,
245             SUM(dt.jumlah_barang) AS total_barang_terjual,
246             AVG(b.harga) AS rata_harga,
247             MIN(b.harga) AS harga_terendah,
248             MAX(b.harga) AS harga_tertinggi
249     FROM transaksi t
250     JOIN detail_transaksi dt ON t.no_transaksi = dt.no_transaksi
251     JOIN barang b ON dt.id_barang = b.id_barang;
252
253
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

jumlah_transaksi	total_barang_terjual	rata_harga	harga_terendah	harga_tertinggi
5	24	55700.000000	3500.00	208000.00

Gambar 3.8: Query Fungsi Agregat Lengkap *serta hasil*

Keterangan :

- COUNT(*) AS jumlah_transaksi
  Digunakan untuk menghitung jumlah seluruh transaksi yang ada di tabel transaksi.
- SUM(dt.jumlah_barang) AS total_barang_terjual
  Digunakan untuk menghitung total seluruh barang yang terjual.
- AVG(b.harga) AS rata_rata_harga
  Digunakan untuk menghitung rata-rata harga barang.
- MIN(b.harga) AS harga_terendah
  Digunakan untuk menampilkan harga barang paling murah.
- MAX(b.harga) AS harga_tertinggi
  Digunakan untuk menampilkan harga barang paling mahal.
- JOIN detail_transaksi dan JOIN barang
  Digunakan untuk menghubungkan tabel transaksi dengan detail transaksi dan data barang agar perhitungan bisa dilakukan.

Hasil Query:

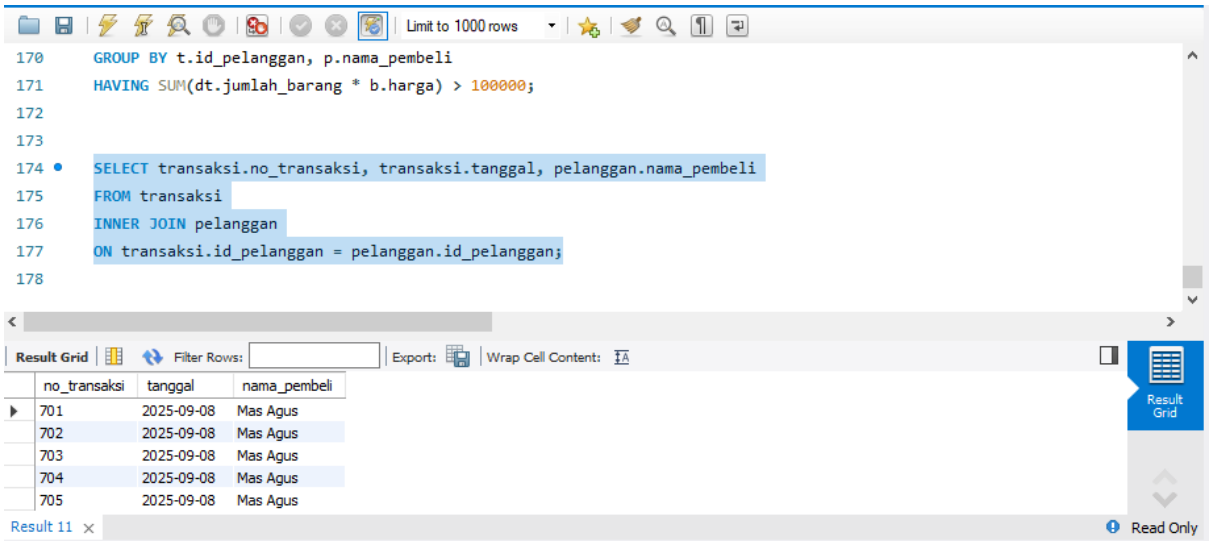
- Berdasarkan hasil yang ditampilkan:
- Jumlah transaksi: 5

- Total barang terjual: 24
- Rata-rata harga barang: Rp 55.700
- Harga terendah: Rp 3.500
- Harga tertinggi: Rp 208.000

4. Implementasi Berbagai Jenis JOIN:

1. INNER JOIN

Query tersebut digunakan untuk menggabungkan data dari dua tabel, yaitu tabel transaksi dan tabel pelanggan, dengan syarat data id_pelanggan pada kedua tabel harus sama.



Gambar 3.9: Query INNER JOIN serta hasil

Keterangan :

- SELECT
  Menentukan kolom yang ingin ditampilkan:
- no_transaksi
- tanggal
- nama_pembeli
- FROM transaksi
  Menentukan tabel utama, yaitu tabel transaksi.
- INNER JOIN pelanggan
  Menggabungkan tabel pelanggan dengan tabel transaksi.
- ON transaksi.id_pelanggan = pelanggan.id_pelanggan
  Syarat penggabungan: data hanya ditampilkan jika id_pelanggan ada di kedua tabel.

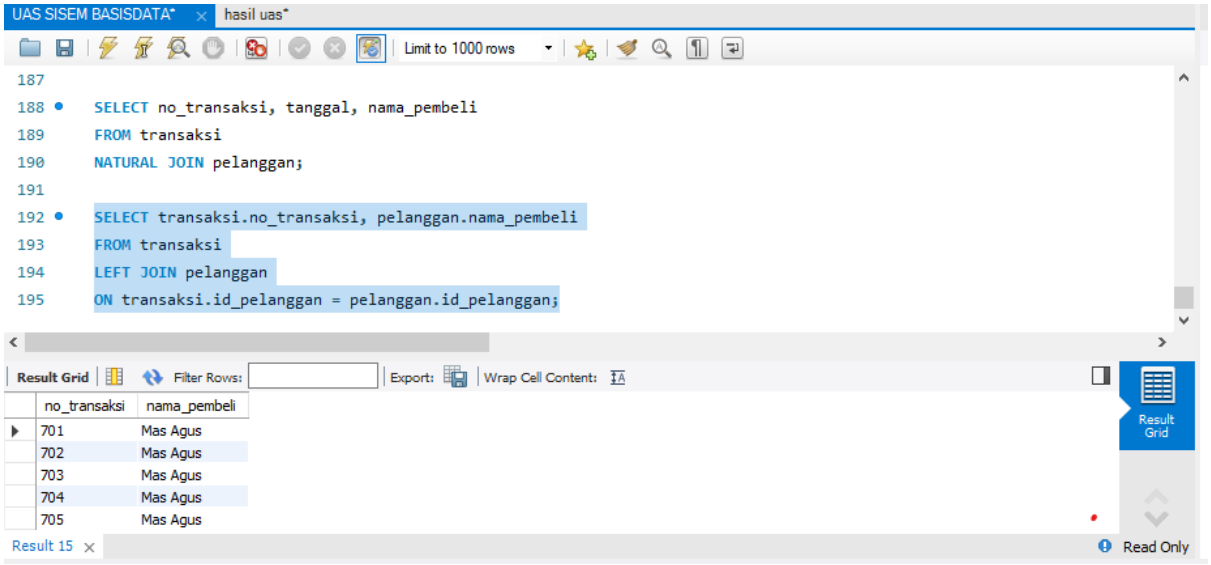
Hasil Query :

- Setiap nomor transaksi memiliki data pelanggan yang cocok.
- Semua transaksi tersebut dilakukan oleh pelanggan bernama Mas Agus.

- Data muncul karena id_pelanggan di tabel transaksi cocok dengan id_pelanggan di tabel pelanggan.

2. LEFT JOIN

Query tersebut digunakan untuk menampilkan seluruh data dari tabel kiri (transaksi),dan data dari tabel kanan (pelanggan) hanya jika ada yang cocok.Jika tidak ada pasangan di tabel kanan, maka kolom dari tabel kanan akan bernilai NULL.



Gambar 4.0: Query LEFT JOIN serta hasil

Keterangan :

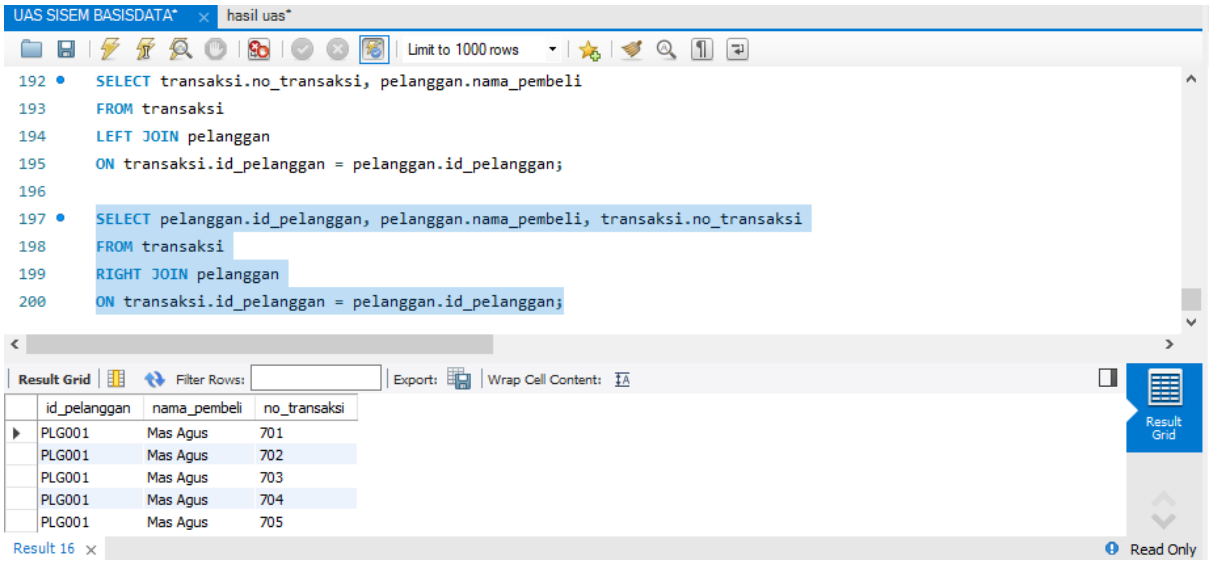
- transaksi → tabel utama (tabel kiri)
- pelanggan → tabel pendukung (tabel kanan)
- LEFT JOIN → semua data dari tabel transaksi pasti ditampilkan
- ON transaksi.id_pelanggan = pelanggan.id_pelanggan
  kondisi relasi antar tabel

Hasil Query :

- Menampilkan seluruh data dari tabel transaksi
- Data dari tabel pelanggan ditampilkan jika id_pelanggan cocok
- Semua transaksi memiliki pasangan data pelanggan
- Kolom nama_pembeli terisi semua (tidak ada nilai NULL)
- Nama pembeli yang muncul adalah Mas Agus untuk setiap transaksi

3. RIGHT JOIN

Query tersebut digunakan untuk menampilkan semua data dari tabel kanan, dan data dari tabel kiri yang memiliki pasangan berdasarkan kondisi ON.Jika tidak ada pasangan di tabel kiri, maka kolom dari tabel kiri akan bernilai NULL.



Gambar 4.1: Query RIGHT JOIN serta hasil

Keterangan:

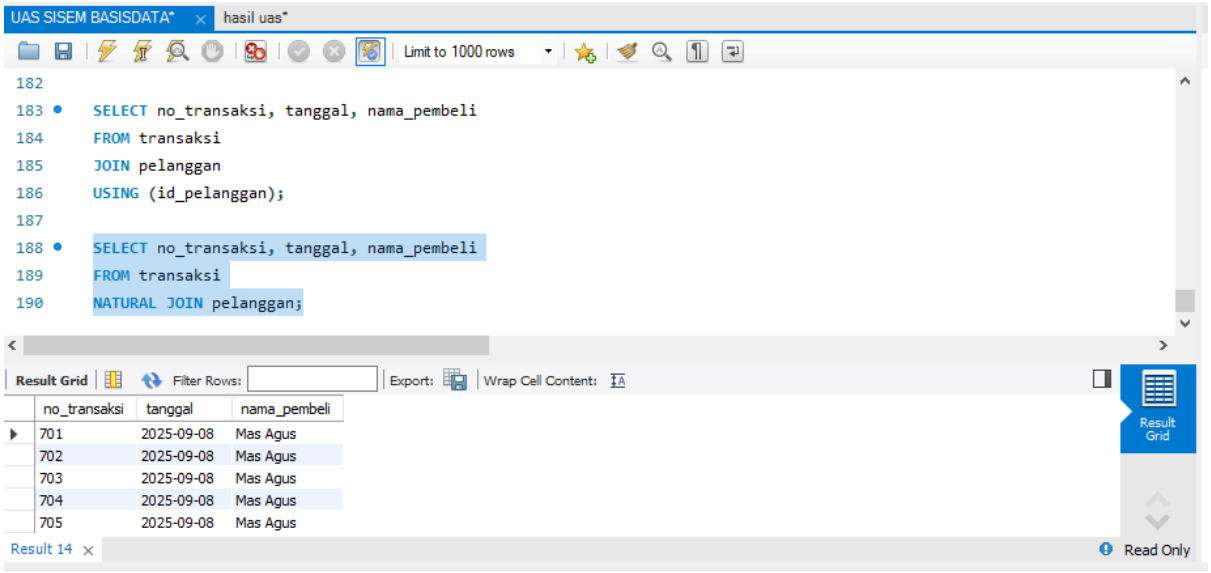
- pelanggan → tabel kanan (semua datanya akan ditampilkan)
- transaksi → tabel kiri
- Data digabung berdasarkan id_pelanggan
- Jika pelanggan belum pernah transaksi, maka no_transaksi akan bernilai NULL

Hasil Query :

- Pelanggan Mas Agus (PLG001) memiliki 5 transaksi
- Karena pelanggan memiliki transaksi, maka kolom no_transaksi terisi
- Jika ada pelanggan lain tanpa transaksi, maka hasilnya akan seperti:

4. NATURAL JOIN

Query tersebut digunakan untuk menggabungkan dua tabel secara otomatis berdasarkan kolom yang memiliki nama yang sama pada kedua tabel tersebut. Pada NATURAL JOIN, pengguna tidak perlu menuliskan kondisi penggabungan menggunakan ON atau USING karena sistem akan langsung mencocokkan kolom yang namanya identik. Jika terdapat data yang cocok pada kolom tersebut, maka baris data akan ditampilkan, sedangkan data yang tidak memiliki kecocokan tidak akan ikut ditampilkan. Selain itu, kolom yang digunakan sebagai penghubung tidak akan ditampilkan secara ganda pada hasil query.



Gambar 4.2: Query NATURAL JOIN serta hasil

Keterangan:

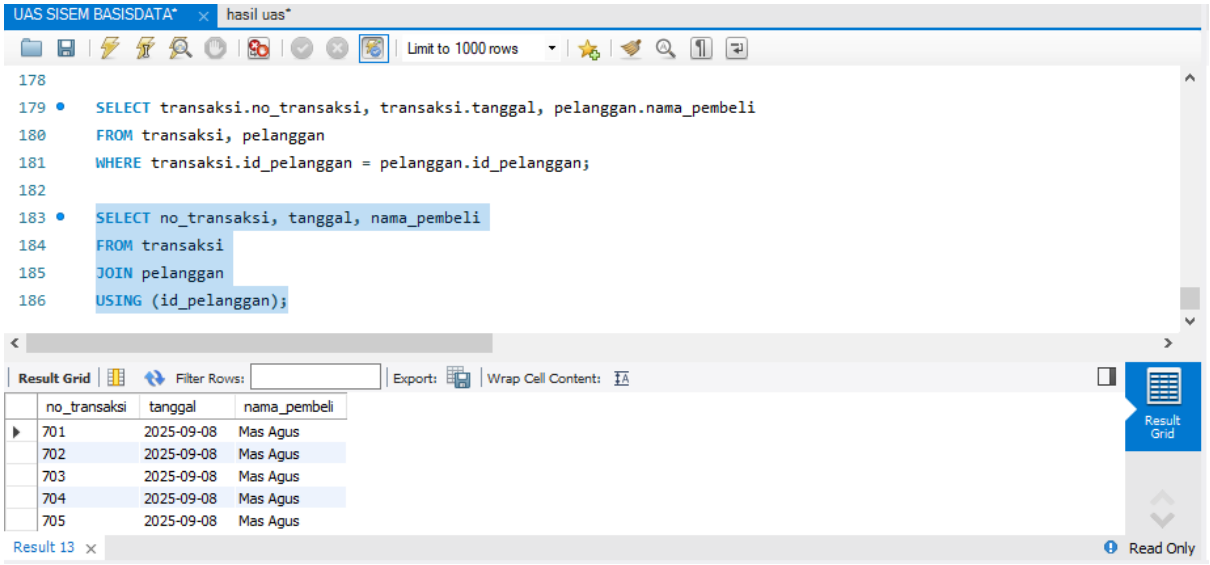
- Menggabungkan tabel secara otomatis
- Berdasarkan kolom yang namanya sama di kedua tabel
- Tidak perlu menuliskan kondisi ON

Hasil Query :

- Setiap transaksi milik Mas Agus
- Data muncul karena id_pelanggan cocok di kedua tabel
- Kolom id_pelanggan tidak ditampilkan otomatis (karakteristik NATURAL JOIN)

5. JOIN DENGAN USING

Query tersebut digunakan untuk penggabungan tabel dalam SQL yang digunakan ketika dua tabel memiliki nama kolom yang sama persis sebagai kunci penghubung (foreign key).
Klausa USING secara otomatis menggabungkan kedua tabel berdasarkan kolom tersebut tanpa perlu menuliskan prefix nama tabel.



Gambar 4.3: Query JOIN DENGAN USING serta hasil

Keterangan:

- Lebih sederhana dan mudah dibaca.
- Tidak perlu prefix tabel jika nama kolom tidak ambigu.
- Hanya perlu menyebutkan kolom kunci di dalam USING().

Hasil Query :

- Semua transaksi (701–705) terjadi pada tanggal 2025-09-08.
- Semua transaksi dilakukan oleh pelanggan yang sama, yaitu Mas Agus.
- Ini menunjukkan bahwa pada tanggal tersebut, Mas Agus melakukan 5 transaksi.

BAB 4 PENUTUP

4.1 Ringkasan Hasil Uji

- Pengujian sistem database menunjukkan hasil yang memuaskan:
- Struktur Database: Semua tabel berhasil dibuat dengan constraint yang tepat
- Integritas Data: Foreign key constraint berfungsi mencegah data orphan
- Query Performance: Semua query berjalan dengan waktu respons yang cepat
- Fungsionalitas: CRUD operations dan query analitis berjalan sesuai ekspektasi

4.2 Kendala dan Perbaikan

- Kendala: Pada tahap awal, terjadi error saat insert data karena urutan tabel yang salah
  Perbaikan: Mengatur urutan INSERT sesuai dependency tabel (parent dulu, kemudian child)
- Kendala: Query dengan multiple JOIN menghasilkan duplikasi data
  Perbaikan: Menggunakan DISTINCT dan merancang ulang kondisi JOIN

4.3 Kesimpulan

Proyek database sistem transaksi toko bangunan telah berhasil diimplementasikan dengan:

- Desain database yang terstruktur dan ternormalisasi hingga 3NF
- Implementasi lengkap DDL, DML, dan query analitis
- Penerapan constraint untuk menjaga integritas data
- Pembuatan berbagai jenis query untuk kebutuhan operasional dan laporan

4.4 Saran Pengembangan

Untuk pengembangan sistem lebih lanjut disarankan:

- Menambahkan tabel supplier untuk manajemen pemasok barang
- Mengimplementasikan stored procedure untuk transaksi kompleks
- Menambahkan trigger untuk audit trail perubahan data
- Membuat view untuk laporan rutin
- Mengimplementasikan indexing untuk optimasi query pada data besar

DAFTAR PUSTAKA

- Connolly T, Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 6th ed. Pearson; 2015.
- MySQL 8.0 Reference Manual. Oracle Corporation; 2024.
- Silberschatz A, Korth HF, Sudarshan S. Database System Concepts. 7th ed. McGraw-Hill; 2020.
- Jurnal Kuliah Pemrograman Basis Data. Universitas Duta Bangsa Surakarta; 2025.