PEMROGAMAN BASIS DATA
NOTA MAKANAN DAN MINUMAN

UNIVERSITAS
DUTA BANGSA
SURAKARTA

Dosen Pengampu Mata Kuliah
Ridwan Dwi Irawan, M.Kom

Disusun Oleh :

Osama Habib Candranata	(240103199)
Samuel Rinlady	(240103202)
Khotijah Naishilla Ariyanto	(240103194)

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS DUTA BANGSA SURAKARTA

TAHUN 2026

KATA PENGANTAR

Puji syukur ke hadirat Allah SWT atas segala limpahan rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan laporan Ujian Akhir Semester mata kuliah Pemrograman Basis Data dengan judul "Perancangan dan Implementasi Basis Data Transaksi di Nota Makanan dan Minuman" dengan baik dan tepat waktu.

Laporan ini disusun sebagai bentuk penerapan materi yang telah dipelajari selama perkuliahan, mulai dari konsep dasar basis data, perancangan Entity Relationship Diagram (ERD), normalisasi, relasi antar tabel, hingga implementasi basis data menggunakan MySQL yang mencakup DDL, DML, TCL, serta query JOIN, GROUP BY, dan HAVING. Studi kasus Natural Digital Printing dipilih untuk memberikan gambaran nyata penerapan sistem basis data pada proses transaksi usaha percetakan.

Penulis menyadari bahwa dalam penyusunan laporan ini masih terdapat keterbatasan dan kekurangan. Oleh karena itu, penulis mengharapkan kritik dan saran yang bersifat membangun demi penyempurnaan laporan ini di masa mendatang. Semoga laporan ini dapat memberikan manfaat, baik sebagai bahan pembelajaran maupun referensi dalam pengembangan sistem basis data.

Akhir kata, penulis mengucapkan terima kasih kepada dosen pengampu mata kuliah Pemrograman Basis Data serta semua pihak yang telah membantu dan mendukung tersusunnya laporan ini.

Surakarta, 17 Januari 2026

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi telah mendorong berbagai sektor usaha kuliner untuk beralih dari sistem manual menuju sistem digital. Sistem penjualan konvensional sering menimbulkan permasalahan seperti duplikasi data, kesalahan perhitungan, dan sulitnya pelacakan stok. Pada studi kasus restoran ini, pencatatan transaksi penjualan makanan—yang mencakup data menu, pelanggan, kasir, dan detail pesanan—masih memiliki risiko tinggi jika tidak dikelola dengan basis data yang terstruktur. Dalam file SQL yang telah dirancang, terlihat kompleksitas hubungan antara stok produk, kategori menu, dan meja makan yang memerlukan penanganan sistematis. Oleh karena itu, diperlukan perancangan basis data relasional yang mampu mengelola integritas data tersebut serta mendukung fitur otomatisasi seperti pengurangan stok otomatis melalui *trigger*.

1.2 Rumusan Masalah

Berdasarkan latar belakang tersebut, rumusan masalah dalam proyek ini adalah:

1. Bagaimana merancang basis data transaksi restoran yang terstruktur dan memenuhi kaidah normalisasi?
2. Bagaimana menerapkan relasi antar tabel (*roles, users, products, orders*) menggunakan *Primary Key* dan *Foreign Key*?
3. Bagaimana mengimplementasikan basis data menggunakan MySQL yang mencakup DDL, DML, dan TCL (transaksi stok)?
4. Bagaimana menampilkan laporan penjualan menggunakan query JOIN dan Aggregasi?

1.3 Tujuan

Adapun tujuan dari penyusunan laporan pemrograman basis data ini adalah sebagai berikut:

1. Merancang ERD (*Entity Relationship Diagram*) untuk sistem restoran.
2. Melakukan normalisasi data dari nota transaksi mentah hingga bentuk 3NF.
3. Mengimplementasikan skema database db_nota_resto beserta *Stored Procedure* dan *Trigger* untuk otomasi.
4. Menghasilkan informasi struk belanja yang akurat melalui query SQL.

1.4 Batasan Masalah

Agar pembahasan dalam proyek ini lebih terfokus, maka batasan masalah ditetapkan sebagai berikut:

1. Sistem mencakup pengelolaan data pengguna (role admin & kasir), meja makan, kategori, produk, pesanan (dine-in/takeaway), dan pembayaran.
2. DBMS yang digunakan adalah MySQL.
3. Fokus pembahasan pada struktur backend database (DDL, DML, TCL, Trigger), bukan antarmuka aplikasi.1.5 Gambaran Umum Sistem

1.5 Gambaran Umum Sistem

Sistem ini mensimulasikan alur restoran dimana pelanggan (atau kasir) melakukan pemesanan (*order*). Sistem akan mencatat detail item, mengurangi stok bahan secara otomatis saat pesanan dibuat, menghitung pajak dan diskon, serta mencetak status pembayaran.

BAB II
LANDASAN TEORI

2.1 Konsep Basis Data dan DBMS

Basis data (database) merupakan kumpulaln informasi yang disimpan dalam komputer dan disusun secara sistematis yang terdiri dari berbagai macam elemen. Secara umum, database digunakan untuk menyimpan data yang telah terintegrasi dengan baik.

Sistem Basis Data adalah system terkomputerisasi yang tujuan utamanya adalah memelihara informasidan membuat informasi tersebut tersedia saat dibutuhkan. Database Management System (DBMS) adalah perangkat lunak yang didesain untuk membantu dalam hal pemeliharaan dan utilitas kumpulan data dalam jumlah besar. DBMS dapat menjadi alternatif penggunaan secara khusus untuk aplikasi, semisal peniympanan data dalam field dan menulis kde aplikasi yang spesifik untuk pemgaturannya [1] [2].

2.2 Relasi Tabel dan Basis Data Relasional

Relasi antar tabel merupakan mekanisme teknis untuk menghubungkan satu tabel dengan tabel lainnya menggunakan primary key dan foreign key. Relasi ini divisualisasikan dalam bentuk diagram relasi yang menggambarkan kolom-kolom kunci dan keterkaitannya. Hardini et al. (2025) menegaskan bahwa penetapan relasi yang tepat antar tabel memastikan konsistensi data dan mendukung proses pengambilan keputusan dalam sistem informasi berbasis relasional.

Basis data relasional adalah sistem yang menggunakan pendekatan berbasis tabel yang saling berhubungan melalui primary key dan foreign key untuk menjaga keterkaitan antar data. SQL memiliki keunggulan pada integritas transaksi dan konsistensi data melalui prinsip ACID (Atomicity, Consistency, Isolation, Durability) yang menjamin keandalan sistem [3] [6].

2.3 Entity Relationship Diagram (ERD)

Entity Relationship Diagram (ERD) adalah representasi visual dari struktur basis data yang menunjukkan hubungan antar entitas, atribut, serta relasi antar tabel. ERD digunakan sebagai alat bantu utama dalam proses perancangan sistem untuk memastikan bahwa model data sesuai dengan kebutuhan fungsional sistem. Menurut Arkan (2025), penggunaan ERD sangat membantu dalam tahap desain karena diagram ini mampu menggambarkan hubungan antar entitas secara intuitif, sehingga memudahkan

pengembang dan pengguna memahami struktur sistem secara keseluruhan.

Selain itu, Saha, Bagui, dan Walsh-Earp (2022) menjelaskan bahwa ERD menjadi alat komunikasi penting antara tim teknis dan non-teknis, membantu meminimalkan kesalahan dalam interpretasi kebutuhan sistem sebelum implementasi dilakukan [4] [5].

2.4 Normalisasi Basis Data

Normalisasi adalah proses sistematis yang bertujuan memperbaiki struktur tabel agar lebih efisien, konsisten, dan bebas dari redundansi data. Melalui normalisasi, setiap atribut data dipastikan hanya memiliki hubungan yang relevan dengan kunci utamanya, sehingga integritas data tetap terjaga. Hardini et al. (2025) menjelaskan bahwa penerapan normalisasi yang konsisten hingga bentuk normal ketiga (3NF) terbukti mampu meningkatkan efisiensi penyimpanan data dan mengurangi risiko duplikasi [3].

2.4.1 Normalisasi 1NF

Tahap pertama normalisasi memastikan setiap atribut berisi nilai tunggal (atomic) dan tidak terdapat kelompok nilai dalam satu kolom. Apabila dalam satu nota terdapat beberapa item barang yang dibeli, maka data tersebut harus dipisahkan ke dalam tabel tersendiri. Menurut Yunianto, Putra, dan Rahmad (2023), penerapan 1NF meningkatkan integritas data serta memudahkan proses pencarian (query) karena struktur tabel menjadi lebih sederhana dan terdefinisi dengan jelas [7] .

2.4.2 Normalisasi 2NF

Tahap kedua bertujuan memastikan bahwa setiap atribut non-key bergantung penuh pada primary key. Dalam kasus di mana tabel memiliki composite key, atribut yang hanya bergantung pada sebagian kunci utama harus dipisahkan ke tabel baru. Dengan cara ini, struktur tabel menjadi lebih fokus dan bebas dari redundansi parsial. Sebagaimana dijelaskan oleh Hardini et al. (2025), pemisahan atribut bergantung parsial menjadi tabel baru meningkatkan fleksibilitas pemeliharaan data serta mengurangi inkonsistensi pada sistem yang kompleks [3].

2.4.3 Normalisasi 3NF

Tahap ketiga normalisasi bertujuan menghapus dependensi transitif, yaitu ketika sebuah atribut non-key bergantung pada atribut non-key lainnya. Dalam bentuk ini, setiap atribut harus bergantung langsung hanya pada primary key utama. Yunianto et al. (2023) menyatakan bahwa struktur 3NF menghasilkan basis data yang lebih efisien dan meminimalkan redundansi antar-entitas, sehingga performa query dan penyimpanan menjadi lebih optimal [7].

2.5 SQL (Structured Query Language)

Menurut Budi, H. S. (2021), SQL (Structured Query Language) adalah sebuah bahasa yang digunakan untuk mengakses data dalam basis data relasional. Bahasa ini secara de facto merupakan bahasa standar ANSI (American Nasional Standard Institute) yang digunakan dalam manajemen basis data relasional yang juga sering disebut dengan istilah query. Saat ini hampir semua server basis data yang ada mendukung bahasa ini untuk melakukan manajemen datanya melalui manipulasi data atau mengedti basis data sesuai yang dikehendaki, seperti menjalankan query untuk mengambil data, menambah data, memperbarui data, dan menghapus data [8]. SQL terbagi menjadi beberapa jenis perintah, antara lain:

2.5.1 Data Definition Language (DDL)

Data Definition Language (DDL) adalah kumpulan perintah SQL yang digunakan untuk mendefinisikan dan mengelola struktur basis data. Perintah DDL digunakan untuk membuat, mengubah, dan menghapus objek basis data seperti database dan tabel [8].
Perintah DDL bersifat permanen, sehingga perubahan yang dilakukan akan langsung tersimpan di dalam basis data.

Contoh perintah yang termasuk dalam DDL antara lain:

1. CREATE DATABASE untuk membuat basis data
2. CREATE TABLE untuk membuat tabel
3. ALTER TABLE untuk mengubah struktur tabel
4. DROP TABLE untuk menghapus tabel

Dalam proyek ini, DDL digunakan untuk membuat database Natural Digital Printing beserta tabel-tabel yang diperlukan, lengkap dengan primary key dan foreign key.

2.5.2 Data Manipulation Language (DML)

Data Manipulation Language (DML) merupakan perintah SQL yang digunakan untuk mengelola data yang tersimpan di dalam tabel. DML berfungsi untuk menambahkan, mengubah, menghapus, dan menampilkan data [8].
Berbeda dengan DDL, perintah DML dapat dikontrol menggunakan transaksi sehingga perubahan data dapat dibatalkan apabila terjadi kesalahan.

Contoh perintah DML antara lain:

1. INSERT untuk menambahkan data
2. UPDATE untuk mengubah data
3. DELETE untuk menghapus data

4. SELECT untuk menampilkan data

Pada proyek ini, DML digunakan untuk mengisikan data pelanggan, barang, karyawan, kasir, serta data transaksi Natural Digital Printing.

2.5.3 Transaction Control Language (TCL)

Transaction Control Language (TCL) adalah kumpulan perintah SQL yang digunakan untuk mengontrol transaksi dalam basis data. TCL berfungsi untuk menjaga konsistensi dan integritas data ketika terjadi proses manipulasi data dalam jumlah besar atau berurutan [8].

Perintah yang termasuk dalam TCL antara lain:

1. START TRANSACTION untuk memulai transaksi
2. COMMIT untuk menyimpan perubahan data secara permanen
3. ROLLBACK untuk membatalkan perubahan data

Dalam proyek ini, TCL digunakan untuk mensimulasikan skenario transaksi, di mana perubahan data dapat dibatalkan apabila terjadi kesalahan input, sehingga integritas data tetap terjaga.

BAB III
PERANCANGAN DAN IMPLEMENTASI

3.1 Analisis Kebutuhan Sistem

a. Sistem ini dirancang untuk mencatat transaksi pembelian makanan. Berdasarkan nota fisik yang dianalisis, terdapat item seperti "Es Kopi Susu Gula Aren" dan "Korean Chicken Skin"

b. Tabel Umum (Unnormalized Table)

Tabel di bawah merupakan hasil gabungan seluruh data dari nota tanpa normalisasi. Artinya, data pelanggan dan transaksi masih berulang di setiap baris karena satu nota berisi lebih dari satu barang. Tabel ini digunakan sebagai dasar awal sebelum dilakukan proses normalisasi, agar nanti bisa dipisahkan menjadi tabel-tabel yang lebih efisien seperti tabel pelanggan, barang, transaksi, dan detail transaksi.

No Nota	**Tanggal**	**Kasir**	**Nama Pelanggan**	**Menu**	**Qty**	**Harga**	**Subtotal**
F-3009	19-10-25	Self Order	Sila	Es Kopi Susu	1	12.900	12.900
F-3009	**19-10-25**	**Self Order**	**Sila**	**Korean Chicken**	**1**	**30.000**	**30.000**

Tabel. 1 Unnormalized Table

c. Normalisasi 1NF (First Normal Form)

Memastikan setiap kolom bernilai atomik (tidak ada pecahan). Data item dipisah per baris seperti tabel di atas..

Transaksi	Id_Pelanggan	Nama Pelanggan	Id_Barang	Nama Barang	Satuan	Banyak Barang	Jumlah Barang
TR001	P001	SILA	1	Es Kopi Susu Gula Aren	pcs	1	12.900
TR002	P002	SILA	2	Korean Chicken Skin	pcs	1	30.000

Tabel. 7 1NF

d. Normalisasi 2NF (Second Normal Form)

Tahap kedua normalisasi bertujuan untuk menghilangkan ketergantungan parsial terhadap kunci utama. Setelah memenuhi 1NF, setiap atribut dalam tabel harus bergantung sepenuhnya pada kunci utama agar tidak terjadi duplikasi atau redundansi data.

1. Tabel Pelanggan

   Tabel Pelanggan di bawah ini sudah memenuhi Second Normal Form (2NF) karena seluruh atribut non-key seperti nama_customer, sepenuhnya bergantung pada satu primary key, yaitu id_pelanggan. Karena kunci utamanya bukan merupakan kunci gabungan, maka tidak mungkin terjadi partial dependency. Selain itu, setiap data dalam tabel sudah bersifat atomik dan tidak memiliki pengulangan, sehingga syarat 1NF juga terpenuhi. Dengan demikian, tabel tersebut dapat dikatakan sudah berada pada bentuk normal 2NF.

ID_PELANGGAN	NAMA PELANGGAN
P001	SILA

Tabel. 8 2NF Pelanggan

2. Tabel Barang

   Tabel Barang berisi informasi mengenai daftar barang yang disimpan atau dijual. Setiap barang diidentifikasi dengan id_barang sebagai penanda unik, misalnya TR001 dan TR002. Kolom nama_barang menjelaskan nama atau jenis barang, seperti ES Kopi Susu Gula Aren dan Korean Chicken Skin. Tabel ini berfungsi untuk mencatat karakteristik dasar dari setiap barang dalam sistem.

ID_BARANG	NAMA	JUMLAH BAYAR	SATUAN
TR001	Es Kopi Susu Gula Aren	12.900	pcs

TR002	Korean Chicken Skin	30.000	pcs

Tabel. 9 2NF Barang

3. Tabel Transaksi

Tabel Order mencatat detail pemesanan barang oleh pelanggan. Setiap pesanan diidentifikasi dengan id_Transaksi, misalnya TR001. Kolom id_transaksi menunjukkan transaksi utama yang terkait dengan pesanan tersebut,qty mencatat jumlah barang yang dipesan, dan harga mencatat harga satuan barang tersebut. Tabel ini berfungsi sebagai pencatat seluruh detail pemesanan dalam sistem.

ID_TRANSAKSI	*QTY*	JUMLAH BAYAR	TANGGAL NOTA	TOTAL
TR001	1	12.900	19/10/2025	42.900

Tabel. 9 2NF Order

4. Detail Transaksi

Sehingga hasil dari 2NF Adalah Struktur tabel semakin rapi dan bebas duplikasi Pemisahan ini membuat struktur data lebih teratur, memudahkan pengelolaan, serta mencegah terjadinya duplikasi data dalam sistem transaksi belanja

e. Normalisasi 3NF (Third Normal Form)

Tahap ini bertujuan untuk menghilangkan ketergantungan antar atribut non-key agar data menjadi lebih efisien dan konsisten. Kolom total harus di hilangkan karena dapat menyebabkan Inconsistency dan Redundancy. Maksud dari Inconsistency dan Redundancy adalah jika detail berubah, total tidak ikut berubah dan data yang sama akan disimpan berkali-kali.

1. Tabel Pelanggan

Tabel Pelanggan menyimpan informasi dasar mengenai pelanggan yang melakukan transaksi. Semua atribut dalam tabel ini sepenuhnya bergantung pada primary key yaitu id_pelanggan, sehingga tidak ada ketergantungan parsial maupun transitive. Struktur tabel ini sudah memenuhi 3NF karena setiap data hanya menjelaskan karakteristik pelanggan dan tidak bergantung pada atribut non-kunci lainnya.

id_pelanggan	nama_customer
P001	Silla

Tabel. 10 3NF Pelanggan

2. Tabel Barang

Tabel Barang berisi data master mengenai barang yang dijual, seperti kode, nama, dan satuan. Seluruh atribut di dalam tabel ini bergantung langsung pada kode_barang sebagai primary key. Tidak terdapat atribut yang saling bergantung satu sama lain sehingga tabel ini sudah sepenuhnya memenuhi 3NF.

Kode_barang	Nama_barang	Ukuran
TR001	Es Kopi Susu Gula Aren	-
TR002	Korean Chicken Skin	-

Tabel. 10 3NF Barang

3. Tabel Transaksi

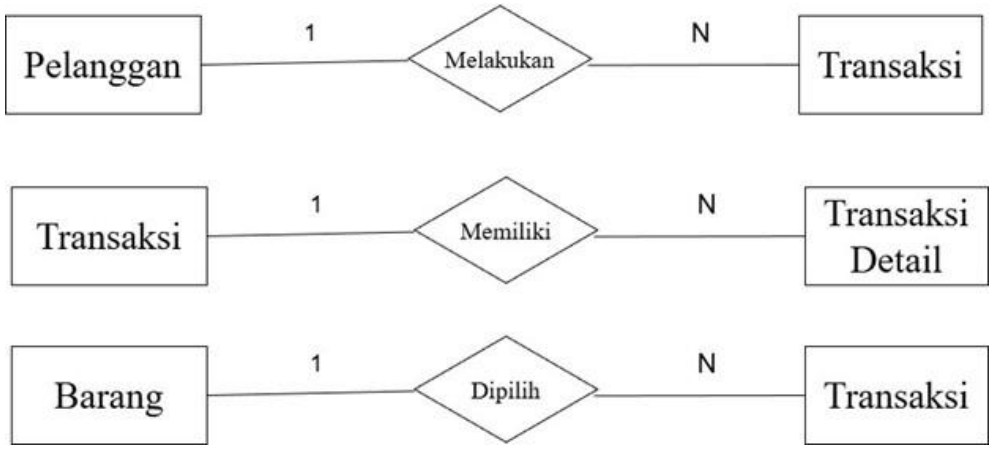
Tabel Transaksi menyimpan informasi utama mengenai transaksi penjualan yang dilakukan oleh pelanggan. Seluruh atribut non-key bergantung langsung pada id_transaksi, dan tidak ada ketergantungan antara atribut non-kunci sehingga terhindar dari anomali transitive. Tabel ini juga berfungsi sebagai penghubung antara pelanggan, kasir, dan detail pembelian, namun tetap mempertahankan prinsip 3NF karena relasinya menggunakan foreign key.

id_transaksi	*QTY*	JUMLAH BAYAR	TANGGAL NOTA	TOTAL
TR01	1	12.900	19/10/2025	42.900

Tabel. 11 3NF Transaksi

f. Derajat Kardinalitas

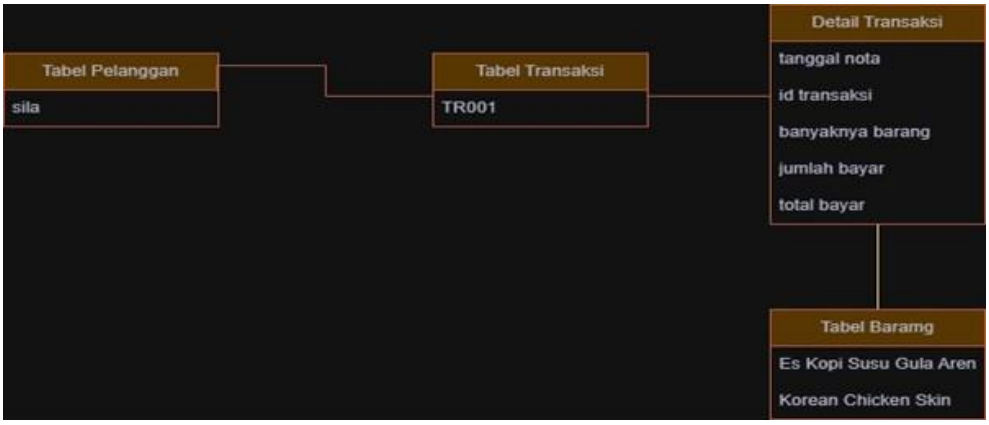
Pada derajat Kardinalitas sendiri menunjukkan hubungan antara dua entitas dalam basis data. Pada data tersebut, setiap hubungan memiliki tipe satu ke (1:N) Dalam system ini memiliki contoh pelanggantransaksi, dalam satu transaksi hanya untuk satu pelanggan. Barang ke Transaksi (1:N) jika transaksi memuat satu barang, maka satu barang dapat muncul di banyak transaksi sedangkan transaksi hanya mencatat satu barang yang di beli. Untuk transaksi ke Barang (N:1) setiap transaksi hanya melibatkan satu barang. Kardinalitas ini memperjelas bagaimana data saling terhubung.



Gambar. 1 Derajat Kardinalitas

g. Relasi Antar Tabel

Relasi antar tabel menunjukkan bahwa tabel Orders (Transaksi) menjadi pusat hubungan data. Satu pelanggan (melalui customer_name), satu barang (produk), dan satu metode pembayaran dapat terhubung ke banyak data transaksi, sedangkan satu kategori dapat menyediakan banyak jenis barang atau produk. Semua hubungan bersifat satu ke banyak (1:N), sehingga data tersimpan lebih rapi dan saling terintegrasi.



Gambar. 2 Relasi Antar Tabel

Diagram tersebut menggambarkan model relasi basis data untuk sistem transaksi penjualan yang terdiri dari entitas utama: pelanggan, transaksi (orders), detail pesanan (order_items), barang (products), karyawan (users), dan kasir.

Entitas pelanggan menyimpan informasi dasar nama pelanggan dan berelasi secara one-to-many dengan entitas transaksi dan order, karena setiap pelanggan dapat melakukan beberapa kali transaksi dan pemesanan.

Entitas transaksi (orders) memiliki atribut terkait detail pembayaran serta status nota dan terhubung dengan entitas kasir (users) serta pelanggan.

Entitas kasir mencatat data operator transaksi dan berelasi dengan transaksi serta peran (roles), menunjukkan bahwa seorang kasir merupakan bagian dari data karyawan dengan hak akses tertentu.

Entitas order_items memuat detail pemesanan seperti barang, kuantitas, dan harga; tabel ini terhubung ke transaksi (orders) dan barang (products), menandakan bahwa setiap baris pesanan berkaitan dengan transaksi tertentu dan berisi barang tertentu.

Entitas barang menyimpan master data produk dan memiliki relasi dengan detail pesanan serta kategori produk.

Sementara itu, entitas karyawan (users) mencatat data pegawai beserta jabatan (roles), serta berhubungan dengan pengelolaan transaksi atau kasir.

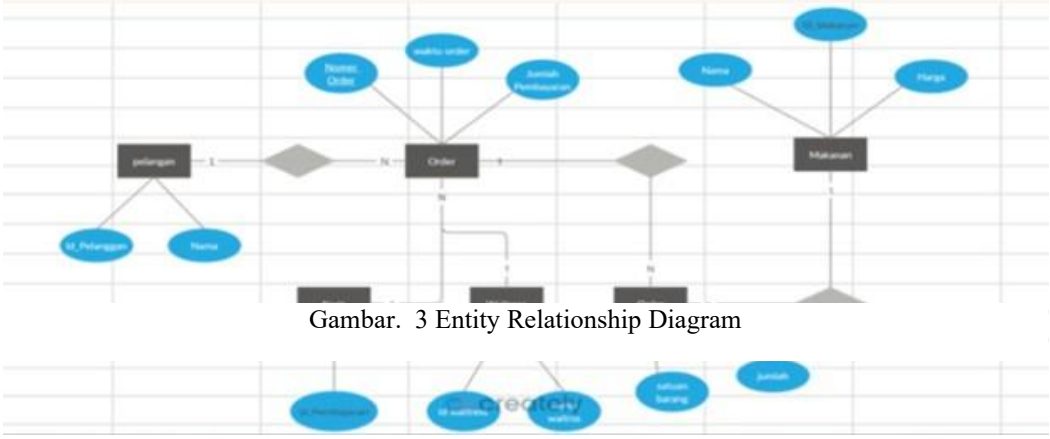
Secara keseluruhan, relasi antar-entitas ini membentuk struktur data yang saling terhubung untuk mendukung proses penjualan, pencatatan transaksi, manajemen pelanggan, serta pengelolaan barang dan karyawan secara terintegrasi.

3.2 Perancangan Database

3.2.1 ERD (Entity Relationship Diagram)

Setelah relasi antar tabel ditentukan, langkah selanjutnya adalah menggambarkan hubungan tersebut dalam bentuk Entity Relationship Diagram (ERD). Diagram ini digunakan untuk memvisualisasikan entitas, atribut, serta jenis hubungan yang terjadi antar entitas. Melalui ERD, rancangan basis data dapat dipahami dengan lebih jelas sebelum diimplementasikan, sehingga hubungan antar tabel dapat terlihat secara menyeluruh dan konsisten.

Diagram Entity–Relationship (ERD) tersebut menggambarkan struktur konseptual sistem informasi penjualan yang terdiri atas entitas Pelanggan, Kasir,



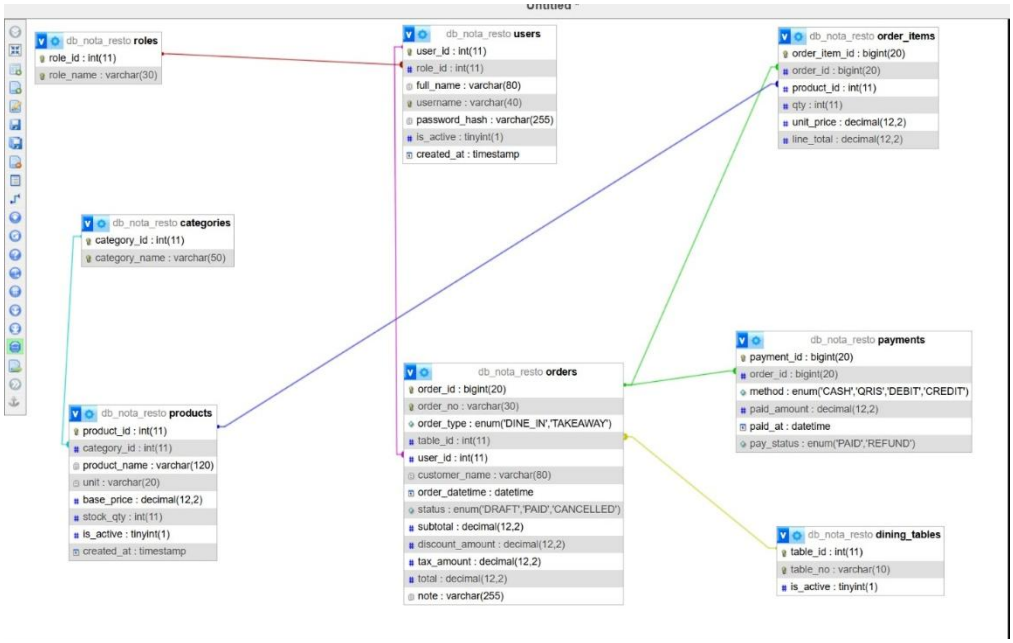
Gambar. 3 Entity Relationship Diagram

Transaksi, Order, dan Barang beserta hubungan antar-entitasnya. Entitas Pelanggan berhubungan dengan entitas Transaksi dan Order melalui relasi melakukan, yang menunjukkan bahwa seorang pelanggan dapat melakukan banyak transaksi serta menghasilkan banyak order. Entitas Order memiliki relasi memiliki dengan entitas Barang, yang mengindikasikan bahwa satu order dapat mencakup banyak barang. Entitas Barang juga terhubung dengan Karyawan melalui relasi mengelola, menandakan bahwa barang tertentu dapat dikelola oleh lebih dari satu karyawan, misalnya dalam konteks pengelolaan inventaris atau penjualan. Sementara itu, entitas Kasir berhubungan dengan Transaksi melalui relasi melayani, menunjukkan bahwa seorang kasir dapat memproses sejumlah transaksi. Relasi-relasi tersebut secara keseluruhan membentuk struktur *many-to-one* dan *one-to-many* yang saling terkait, sehingga mencerminkan alur bisnis mulai dari interaksi pelanggan, pencatatan order, pengelolaan barang, hingga pemrosesan transaksi oleh kasir..

3.2.2 Skema Tabel

Skema tabel basis data pada sistem **Nota Makanan dan Minuman (db_nota_resto)** dirancang untuk mendukung proses pengelolaan data produk, pengguna (kasir/admin), meja restoran, hingga transaksi penjualan serta detail pembayaran secara terintegrasi. Perancangan skema tabel ini bertujuan untuk memastikan struktur data yang terorganisir, menghindari redundansi data, serta menjaga konsistensi hubungan antar tabel melalui penerapan *primary key* dan *foreign key*. Seluruh tabel saling terhubung sesuai dengan kebutuhan proses bisnis, mulai dari pengaturan kategori produk hingga pencatatan detail pembayaran pelanggan.

Untuk mempermudah pemahaman terhadap struktur basis data, seluruh skema tabel ditampilkan dalam bentuk *screenshot* yang menunjukkan susunan kolom, tipe data, serta relasi antar tabel. Adapun ringkasan skema tabel yang digunakan dalam sistem database restoran ini dapat dilihat pada gambar berikut:



Gambar. 4 Skema Relasi Tabel Basis Data

Gambar di atas menunjukkan skema relasi tabel pada basis data Nota Restoran yang terdiri dari tabel roles, users, categories, products, dining_tables, orders, order_items, dan payments. Setiap tabel memiliki primary key sebagai identitas unik, sedangkan hubungan antar tabel dibentuk menggunakan foreign key untuk menjaga integritas data. Tabel orders berperan sebagai tabel utama yang menghubungkan data pelanggan (melalui nama pelanggan), kasir (user), dan meja restoran, sedangkan tabel order_items berfungsi sebagai tabel detail yang mencatat setiap item produk dalam sebuah transaksi. Tabel payments melengkapi siklus transaksi dengan mencatat metode dan status pembayaran. Relasi antar tabel ini dirancang untuk mendukung proses pencatatan transaksi penjualan secara terstruktur, konsisten, dan akurat.

3.2.3 PK & FK

Primary Key (PK) adalah kunci utama dalam sebuah tabel basis data yang berfungsi sebagai identitas unik untuk setiap data. PK digunakan untuk membedakan satu record dengan record lainnya sehingga tidak terjadi data ganda. Nilai Primary Key harus bersifat unik dan tidak boleh kosong. Setiap tabel hanya memiliki satu Primary Key, dan kunci ini sangat penting untuk memudahkan pencarian data serta menjadi acuan dalam pembentukan relasi antar tabel.

Foreign Key (FK) adalah kunci yang digunakan untuk menghubungkan satu tabel dengan tabel lainnya dalam basis data. FK merupakan kolom yang nilainya mengacu pada Primary Key di tabel lain. Foreign Key berfungsi untuk menjaga keterkaitan dan konsistensi data antar tabel, sehingga data yang dimasukkan harus sesuai dengan data yang sudah ada pada tabel yang dirujuk. Dengan adanya Foreign Key, hubungan antar data menjadi terstruktur dan mencegah terjadinya kesalahan atau ketidaksesuaian data.

a. Primary Key

Primary Key adalah kolom yang mengidentifikasi setiap record secara unik dalam tabel. Setiap tabel hanya memiliki satu Primary Key dengan nilai yang tidak boleh NULL atau duplikat. Fungsinya adalah menjamin keunikan data, menjadi acuan untuk Foreign Key, mempercepat pencarian data, dan mencegah duplikasi.

Database ini memiliki 8 Primary Key yaitu role_id pada tabel roles, user_id pada tabel users, table_id pada tabel dining_tables, category_id pada tabel categories, product_id pada tabel products, order_id pada tabel orders, order_item_id pada tabel order_items, dan payment_id pada tabel payments.

Tabel	Primay Key	Deskripsi
roles	role_id	ID unik untuk setiap role (admin, kasir)
User	user_id	ID unik untuk setiap user/pengguna sistem

dining_tables	table_id	ID unik untuk setiap meja makan
categories	category_id	ID unik untuk setiap kategori produk (Makanan, Minuman, Snack)
products	product_id	ID unik untuk setiap produk/menu
orders	order_id	ID unik untuk setiap nota/pesanan
order_items	order_item_id	ID unik untuk setiap item dalam pesanan
payments	payment_id	ID unik untuk setiap transaksi pembayaran

Tabel. 2 Primary Key

b. Foreign Key

Foreign Key adalah kolom yang merujuk ke Primary Key di tabel lain untuk membuat relasi antar tabel. Fungsinya adalah menghubungkan data dari tabel berbeda, menjaga integritas data, mencegah data orphan, dan mengatur aksi otomatis. Constraint yang digunakan meliputi ON DELETE CASCADE (hapus otomatis), ON DELETE RESTRICT (cegah penghapusan), ON DELETE SET NULL (set NULL), dan ON UPDATE CASCADE (update otomatis).

Database ini memiliki 7 Foreign Key. Tabel users memiliki role_id yang mereferensi roles(role_id). Tabel products memiliki category_id yang mereferensi categories(category_id). Tabel orders memiliki table_id yang mereferensi dining_tables(table_id) dan user_id yang mereferensi users(user_id). Tabel order_items memiliki order_id yang mereferensi orders(order_id) dan product_id yang mereferensi products(product_id). Tabel payments memiliki order_id yang mereferensi orders(order_id).

Relasi yang terbentuk adalah One-to-Many di mana satu record induk dapat memiliki banyak record anak, seperti satu role untuk banyak users, satu order untuk banyak order_items dan payments.

Tabel	FK Column	Referensi	Constraint	Deskripsi
users	role_id	roles(role_id)	fk_users_role	**Menghubungkan user dengan role-nya. ON UPDATE CASCADE, ON DELETE RESTRICT (tidak bisa hapus role yang masih dipakai)**
products	category_id	categories(category_id)	fk_products_category	Menghubungkan produk dengan kategorinya. ON UPDATE CASCADE, ON DELETE RESTRICT (tidak bisa hapus kategori yang masih punya produk)
orders	table_id	dining_tables(table_id)	fk_orders_table	Menghubungkan order dengan meja (untuk DINE_IN). ON UPDATE CASCADE, ON DELETE SET NULL (jadi NULL jika meja dihapus)

Order	Id_transaksi	transaksi	Id_transaksi	Menghubungkan order dengan meja (untuk DINE_IN). ON UPDATE CASCADE, ON DELETE SET NULL (jadi NULL jika meja dihapus)
orders	user_id	users(user_id)	fk_orders_user	Menghubungkan order dengan kasir yang menginput. ON UPDATE CASCADE, ON DELETE RESTRICT (tidak bisa hapus user yang pernah buat transaksi)
order_items	order_id	orders(order_id)	fk_items_order	Menghubungkan item dengan nota pesanannya. ON UPDATE CASCADE, ON DELETE CASCADE (item ikut terhapus jika order dihapus)
order_items	product_id	products(product_id)	fk_items_product	Menghubungkan item dengan produk yang dipesan. ON UPDATE CASCADE, ON DELETE RESTRICT

				(tidak bisa hapus produk yang ada di transaksi)
payments	order_id	orders(order_id)	fk_payments_order	Menghubungkan pembayaran dengan nota yang dibayar. ON UPDATE CASCADE, ON DELETE CASCADE (pembayaran ikut terhapus jika order dihapus)

Tabel. 3 Foreign Key

3.3 Implementasi Basis Data

Implementasi basis data pada Nota Makanan Dan Minuman dilakukan secara langsung menggunakan DBMS MySQL berdasarkan hasil perancangan yang telah dibuat pada tahap sebelumnya. Implementasi ini mencakup penerapan **Data Definition Language (DDL)** untuk membangun struktur basis data, **Data Manipulation Language (DML)** untuk pengisian dan pengelolaan data, serta **Transaction Control Language (TCL)** untuk pengujian mekanisme transaksi. Seluruh proses implementasi telah dijalankan dan dibuktikan melalui pengujian langsung pada basis data.

3.3.1 Implementasi DDL

Implementasi DDL dilakukan untuk membangun struktur basis data sesuai dengan rancangan Nota Makanan Dan Minuman Pada tahap ini, database dan seluruh tabel utama berhasil dibuat, meliputi tabel pelanggan, karyawan, kasir, barang, transaksi, dan order. Setiap tabel telah dilengkapi dengan **Primary Key** sebagai identitas unik data serta **Foreign Key** untuk membentuk relasi antar tabel. Selain itu, penerapan constraint seperti **NOT NULL**, **ON UPDATE CASCADE**, dan **ON DELETE RESTRICT/CASCADE** telah diuji dan berfungsi dengan baik dalam menjaga integritas data. Keberhasilan implementasi DDL dibuktikan dengan terbentuknya seluruh tabel dan relasi sesuai perancangan. Script DDL ditampilkan pada bagian lampiran 1.

3.3.2 Implementasi DML

Implementasi DML dilakukan untuk mengisi dan memanipulasi data pada basis data Nota Makanan Dan Minuman. Pengujian DML meliputi proses penambahan data pelanggan, karyawan, kasir, barang, transaksi, serta detail order menggunakan perintah INSERT. Selain itu, perintah SELECT digunakan untuk memverifikasi bahwa data berhasil tersimpan dan relasi antar tabel berjalan dengan benar. Hasil implementasi menunjukkan bahwa seluruh data dapat disimpan, ditampilkan, dan saling terhubung sesuai dengan aturan Foreign Key yang diterapkan. Script DML lengkap disajikan pada bagian lampiran 2.

3.3.3 Implementasi TCL

Implementasi Transaction Control Language (TCL) dilakukan untuk menguji pengendalian transaksi dalam basis data. Pengujian dilakukan dengan menjalankan skenario transaksi menggunakan perintah START TRANSACTION, diikuti dengan proses INSERT data transaksi, kemudian dilakukan pembatalan menggunakan perintah ROLLBACK. Hasil pengujian menunjukkan bahwa data yang dibatalkan tidak tersimpan dalam basis data, sehingga membuktikan bahwa mekanisme pengendalian transaksi berjalan dengan baik dan konsistensi data tetap terjaga. Script TCL ditampilkan pada bagian lampiran 3.

3.4 Implementasi Query SQL

Implementasi query SQL pada sistem Nota Makanan Dan Minuman dilakukan untuk menguji kemampuan basis data dalam menghasilkan informasi yang dibutuhkan sesuai dengan proses bisnis sistem. Query SQL yang diimplementasikan meliputi query **JOIN**, **GROUP BY**, **HAVING**, serta penggunaan fungsi **agregasi** seperti SUM, COUNT, AVG, MAX, dan MIN. Seluruh query dijalankan langsung pada basis data yang telah diisi dengan data uji.Implementasi Query Sql dari JOIN sampai Agregasi dapat dilihat di lampiran 4

3.4.1 Query JOIN

Implementasi query JOIN dilakukan untuk menggabungkan data dari beberapa tabel yang saling berelasi, yaitu tabel order, transaksi, pelanggan, dan barang. Pengujian JOIN menunjukkan bahwa data dapat ditampilkan secara terintegrasi, sehingga informasi detail pesanan seperti nama pelanggan, nomor bukti transaksi, nama barang, jumlah, harga, dan subtotal dapat ditampilkan dalam satu hasil query. Keberhasilan implementasi ini membuktikan bahwa relasi antar tabel telah berjalan dengan baik sesuai dengan perancangan. Script query JOIN ditampilkan pada bagian lampiran.

3.4.2 Query GROUP BY

Implementasi query GROUP BY dilakukan untuk mengelompokkan data berdasarkan identitas transaksi dan menghitung total nilai transaksi. Pada pengujian ini, fungsi agregasi SUM digunakan untuk menjumlahkan nilai subtotal dari setiap item dalam transaksi yang sama. Hasil pengujian menunjukkan bahwa total transaksi yang dihasilkan sesuai dengan nilai total_harga pada tabel transaksi, sehingga membuktikan keakuratan perhitungan dan konsistensi data. Script query GROUP BY disajikan pada bagian lampiran.

3.4.3 Query HAVING

Implementasi query HAVING dilakukan untuk memfilter hasil data yang telah dikelompokkan berdasarkan kondisi tertentu. Pada sistem ini, HAVING digunakan untuk menampilkan transaksi dengan total nilai di atas batas tertentu. Hasil pengujian menunjukkan bahwa hanya transaksi yang memenuhi kondisi yang ditampilkan, sehingga membuktikan bahwa query HAVING berfungsi dengan baik dalam menyaring hasil agregasi. Script query HAVING ditampilkan pada bagian lampiran.

3.4.4 Query Agregasi

Implementasi query agregasi dilakukan untuk menganalisis data penjualan dalam basis data Natural Digital Printing. Fungsi agregasi seperti SUM digunakan untuk menghitung total nilai transaksi, COUNT untuk menghitung jumlah transaksi, AVG untuk menghitung rata-rata nilai transaksi, serta MAX dan MIN untuk mengetahui transaksi dengan nilai tertinggi dan terendah. Hasil pengujian menunjukkan bahwa query agregasi mampu menghasilkan informasi statistik yang akurat dan berguna sebagai bahan analisis data penjualan. Script query agregasi lengkap disajikan pada bagian lampiran.

BAB IV
PENUTUP

4.1 Hasil Pengujian

Hasil pengujian basis data pada Nota Makanan Dan Minuman menunjukkan bahwa seluruh perintah **Data Definition Language (DDL)**, **Data Manipulation Language (DML)**, dan **Transaction Control Language (TCL)** dapat dijalankan dengan baik dan sesuai dengan perancangan sistem. Pengujian DDL membuktikan bahwa database serta seluruh tabel berhasil dibuat tanpa error, lengkap dengan penerapan Primary Key dan Foreign Key yang berfungsi sesuai aturan **CASCADE** dan **RESTRICT**, sehingga integritas data antar tabel tetap terjaga. Selanjutnya, pengujian DML menunjukkan bahwa proses penambahan dan penampilan data pada setiap tabel berjalan dengan benar, di mana data pelanggan, karyawan, kasir, barang, transaksi, dan detail order berhasil disimpan serta saling terhubung sesuai relasinya. Sementara itu, pengujian TCL memperlihatkan bahwa mekanisme pengendalian transaksi berjalan secara optimal, ditandai dengan keberhasilan proses **ROLLBACK** dalam membatalkan penyimpanan data transaksi yang tidak dikonfirmasi. Secara keseluruhan, hasil pengujian ini menunjukkan bahwa basis data telah berfungsi secara konsisten, aman, dan siap digunakan untuk mendukung operasional sistem Natural Digital Printing.

Objek Uji	Perintah SQL	Skenario Pengujian	Hasil yang diharapkan	Hasil aktual	Status
Database	CREATE DATABASE db_nota_resto	Membuat database baru untuk sistem nota restoran.	Database db_nota_resto berhasil dibuat	Database berhasil dibuat dan dapat digunakan.	Berhasil
Tabel Roles	CREATE TABLE roles (...)	Membuat tabel master untuk menyimpan tingkatan user (admin/kasir).	Tabel roles terbentuk dengan role_id sebagai Primary Key.	Tabel terbentuk sesuai skema.	Berhasil

Tabel Users	CREATE TABLE users (...)	Membuat tabel user dengan Foreign Key ke tabel roles karyawan dengan PK	Tabel users terbentuk; kolom username bersifat unik.	Tabel terbentuk; relasi FK ke roles aktif.	Berhasil
Tabel Dining Tables	CREATE TABLE dining_tables (...)	Membuat tabel untuk daftar meja restoran.	Tabel dining_tables terbentuk dengan kolom table_no.	Tabel terbentuk dengan benar.	Berhasil
Tabel Categories	CREATE TABLE categories (...)	Membuat tabel kategori produk (makanan/minuman).	Tabel categories terbentuk.	Tabel terbentuk dengan benar.	Berhasil
Tabel Products	CREATE TABLE products (...)	Membuat tabel produk dengan constraint CHECK pada harga dan stok.	Tabel products terbentuk; harga dan stok tidak boleh negatif.	Tabel terbentuk; constraint CHECK aktif.	Berhasil
Tabel Orders	CREATE TABLE orders (...)	Membuat tabel transaksi (nota) dengan tipe data ENUM dan relasi ke user/meja.	Tabel orders terbentuk; mendukung tipe DINE_IN dan TAKEAWAY.	Tabel terbentuk dengan kolom ENUM yang sesuai.	Berhasil
Tabel Order Items	CREATE TABLE order_items (...)	Membuat tabel detail item per nota dengan relasi	Tabel terbentuk; jika order dihapus,	Tabel terbentuk; relasi CASCA	Berhasil

		ON DELETE CASCADE.	item di dalamnya ikut terhapus.	DE berfungsi.	
Tabel Payments	CREATE TABLE payments (...)	Membuat tabel pembayaran dengan metode ENUM (CASH, QRIS, dll).	Tabel payments terbentuk untuk mencatat riwayat bayar.	Tabel terbentuk dengan benar.	Berhasil
Index	CREATE INDEX idx_products_category ON products(category_id)	Membuat indeks pada kolom yang sering digunakan untuk pencarian/join	Pencarian produk berdasarkan kategori menjadi lebih cepat.	Indeks berhasil dibuat di tabel products dan orders.	Berhasil

Tabel. 4 Pengujian DDL

Objek Uji	Perintah SQL	Data Uji	Hasil yang diharapkan	Hasil aktual	Status
Insert Master Data	INSERT INTO products (...)	Produk: 'Kopi Susu', Harga: 12000, Stok: 60	Data tersimpan di tabel products.	Data tersimpan dengan product_id otomatis.	Berhasil
Stored Procedure (Transaction)	CALL sp_create_order_demo (...)	Order 'INV-0001', Tipe 'DINE_IN', User ID 2, Diskon 2000	Transaksi tersimpan ke tabel orders dan order_items secara atomik.	Data tersimpan; subtotal dan total terhitung otomatis.	Berhasil

Trigger: Potong Stok	AFTER INSERT ON order_items	Insert 2 porsi 'Nasi Goreng Spesial' (Stok awal: 50)	Stok produk berkurang otomatis menjadi 48.	Stok berkurang menjadi 48 setelah prosedur dijalankan.	Berhasil
Trigger: Validasi Stok	BEFORE INSERT ON order_items	Pesan 100 porsi 'Ayam Geprek' (Stok tersedia: 35)	Muncul error: 'Stok tidak cukup untuk item yang ditambahkan'.	Pesan error muncul dan transaksi dibatalkan (Rollback).	Berhasil
Trigger: Update Stok	AFTER UPDATE ON order_items	Mengubah Qty order dari 2 menjadi 5	Stok menyesuaikan selisih (berkurang lagi 3).	Stok produk terupdate secara akurat sesuai selisih.	Berhasil
Update Status Pembayara	UPDATE orders SET status = 'PAID' ...	Order 'INV-0001'	Status berubah dari 'DRAFT' menjadi 'PAID'.	Kolom status terupdate menjadi 'PAID'.	Berhasil
Aggregated Query (Join)	SELECT SUM(total) ... GROUP BY DATE	Data transaksi 'INV-0001' dan 'INV-0002'	Menampilkan total penjualan harian yang benar.	Hasil sesuai dengan total perhitungan manual.	Berhasil

Subquery Validation	SELECT ... WHERE total > (SELECT AVG(total)...)	Daftar seluruh transaksi	Menampilkan nota dengan nilai di atas rata-rata.	Hanya transaksi bernilai besar yang muncul.	Berhasil

Tabel. 5 Pengujian DML

Urutan Uji	Perintah SQL	Skenario	Hasil yang diharapkan	Hasil aktual	Status
1	START TRANSACTION	Memulai sesi transaksi di dalam prosedur sp_create_order_demo sebelum melakukan input data.	Sistem menandai dimulainya blok transaksi yang bersifat sementara (buffer).	Sesi transaksi dimulai dengan sukses.	Berhasil
2	INSERT INTO orders & order_items	Menginput data header nota dan detail item pesanan di tengah transaksi yang masih berjalan.	Data tersimpan secara sementara namun belum terlihat oleh pengguna lain sebelum di-commit.	Data masuk ke tabel tetapi masih dalam status "pending" di level database.	Berhasil
3	COMMIT	Menjalankan perintah COMMIT setelah seluruh proses input dan perhitungan pajak/total selesai	Semua perubahan data disimpan secara permanen	Data nota dan item tersimpan permanen dan stok produk terpotong	Berhasil

			ke dalam database.	secara resmi.	
4	ROLLBACK (Simulasi Gagal)	Melakukan input item dengan jumlah yang melebihi stok (memicu error pada trigger).	Sistem otomatis membatalkan seluruh rangkaian insert (termasuk header nota) agar tidak ada data sampah.	Transaksi dibatalkan secara keseluruhan; tidak ada data parsial yang tersimpan.	Berhasil
5	LAST_INSERT_ID()	Mengambil ID dari tabel orders untuk digunakan sebagai Foreign Key pada tabel order_items dalam satu transaksi.	ID yang didapat harus akurat sesuai dengan baris yang baru saja dibuat dalam sesi tersebut.	ID berhasil didapatkan dan relasi antar tabel terbentuk dengan benar.	Berhasil

Tabel. 6 Pengujian TCL

4.2 Kendala dan Solusi

Salah satu kendala utama yang ditemukan adalah adanya risiko ketidaksinkronan data ketika terjadi kegagalan sistem di tengah proses input nota dan item pesanan, namun hal ini berhasil diatasi dengan menerapkan TCL (Transaction Control Language) melalui perintah START TRANSACTION dan COMMIT di dalam Stored Procedure guna menjamin seluruh data tersimpan secara utuh.

Ditemukan pula potensi terjadinya stok minus apabila beberapa transaksi dilakukan secara bersamaan atau jika jumlah pesanan melebihi stok yang tersedia, sehingga diimplementasikan Trigger BEFORE INSERT yang secara otomatis melakukan pengecekan stok dan membatalkan transaksi menggunakan SIGNAL SQLSTATE jika stok tidak mencukupi.

Muncul kendala dalam menjaga akurasi stok saat kasir melakukan perubahan jumlah atau penghapusan item pada nota yang sudah ada, yang kemudian diselesaikan dengan pembuatan Trigger AFTER UPDATE dan AFTER DELETE untuk melakukan penyesuaian atau pengembalian stok ke tabel produk secara otomatis.

Proses perhitungan nilai subtotal, pajak, dan diskon pada nota awalnya rawan terhadap kesalahan manusia jika dilakukan secara manual, sehingga solusi yang diambil adalah mengintegrasikan logika perhitungan aritmatika langsung di dalam Stored Procedure agar sistem dapat menghitung total bayar secara otomatis dan akurat.

4.3 Kesimpulan

Berdasarkan hasil perancangan, implementasi, dan pengujian basis data pada sistem Nota Restoran, dapat disimpulkan bahwa proses perancangan basis data transaksi telah berhasil dilakukan sesuai dengan kebutuhan sistem. Perancangan basis data dimulai dari analisis kebutuhan hingga penyusunan tabel yang terstruktur, sehingga basis data menjadi lebih efisien, konsisten, dan bebas dari redundansi data.

Implementasi basis data menggunakan DBMS MySQL berhasil dilakukan dengan menerapkan perintah Data Definition Language (DDL) untuk membangun struktur database, Data Manipulation Language (DML) untuk mengelola data, serta Transaction Control Language (TCL) untuk mengendalikan transaksi. Penerapan Primary Key dan Foreign Key pada setiap tabel, seperti relasi antara tabel produk dan kategori serta nota dan item pesanan, mampu menjaga integritas data dan memastikan keterkaitan antar tabel berjalan dengan baik sesuai dengan aturan relasi yang telah dirancang.

Selain itu, pengujian query SQL seperti JOIN untuk detail invoice, GROUP BY dan fungsi agregasi untuk laporan harian, serta penggunaan HAVING untuk filter penjualan produk menunjukkan bahwa basis data mampu menghasilkan informasi transaksi secara akurat dan terstruktur. Dengan demikian, sistem basis data yang dibangun telah memenuhi tujuan proyek, yaitu mendukung pengelolaan data kategori, produk, meja, pengguna (admin/kasir), serta transaksi penjualan secara terintegrasi pada sistem nota restoran.

4.4 Saran

Berdasarkan hasil pengembangan dan pengujian sistem basis data ini, terdapat beberapa saran yang dapat dipertimbangkan untuk pengembangan selanjutnya. Sistem basis data dapat dikembangkan lebih lanjut dengan menambahkan antarmuka aplikasi

(*User Interface*) agar proses input data pesanan dan pengelolaan stok menjadi lebih mudah bagi pengguna. Selain itu, dapat ditambahkan fitur keamanan data yang lebih mendalam, seperti penguatan pengaturan hak akses pengguna melalui tabel roles untuk membedakan secara ketat wewenang antara admin, kasir, dan manajer guna meningkatkan kontrol informasi.

Pengembangan selanjutnya juga dapat mencakup penambahan modul laporan otomatis yang lebih komprehensif, seperti laporan penjualan harian, bulanan, dan tahunan yang memanfaatkan *View* v_invoice_detail, serta integrasi lebih lanjut dengan manajemen stok barang yang mencatat setiap mutasi secara kronologis. Dengan pengembangan tersebut, sistem basis data nota restoran ini diharapkan dapat memberikan manfaat yang lebih optimal dan mendukung operasional usaha secara lebih efisien dan profesional.

LAMPIRAN

Lampiran 1
Script SQL Data Definition Language (DDL)

```
/* ============================================================

   DDL – DATA DEFINITION LANGUAGE

   ============================================================ */


-- DATABASE

DROP DATABASE IF EXISTS db_nota_resto;

CREATE DATABASE db_nota_resto

  CHARACTER SET utf8mb4

  COLLATE utf8mb4_unicode_ci;


USE db_nota_resto;


-- ============================================================

-- TABEL MASTER

-- ============================================================


CREATE TABLE roles (

  role_id INT AUTO_INCREMENT PRIMARY KEY,

  role_name VARCHAR(30) NOT NULL UNIQUE

) ENGINE=InnoDB;
```

```
CREATE TABLE users (
  user_id INT AUTO_INCREMENT PRIMARY KEY,
  role_id INT NOT NULL,
  full_name VARCHAR(80) NOT NULL,
  username VARCHAR(40) NOT NULL UNIQUE,
  password_hash VARCHAR(255) NOT NULL,
  is_active TINYINT(1) NOT NULL DEFAULT 1,
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_users_role
    FOREIGN KEY (role_id) REFERENCES roles(role_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=InnoDB;

CREATE TABLE dining_tables (
  table_id INT AUTO_INCREMENT PRIMARY KEY,
  table_no VARCHAR(10) NOT NULL UNIQUE,
  is_active TINYINT(1) NOT NULL DEFAULT 1
) ENGINE=InnoDB;

CREATE TABLE categories (
  category_id INT AUTO_INCREMENT PRIMARY KEY,
  category_name VARCHAR(50) NOT NULL UNIQUE
) ENGINE=InnoDB;
```

```
CREATE TABLE products (
  product_id INT AUTO_INCREMENT PRIMARY KEY,
  category_id INT NOT NULL,
  product_name VARCHAR(120) NOT NULL,
  unit VARCHAR(20) NOT NULL DEFAULT 'porsi',
  base_price DECIMAL(12,2) NOT NULL CHECK (base_price >= 0),
  stock_qty INT NOT NULL DEFAULT 0 CHECK (stock_qty >= 0),
  is_active TINYINT(1) NOT NULL DEFAULT 1,
  created_at TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
  CONSTRAINT fk_products_category
    FOREIGN KEY (category_id) REFERENCES categories(category_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
) ENGINE=InnoDB;

CREATE INDEX idx_products_category
ON products(category_id);

-- ============================================================
-- TABEL TRANSAKSI
-- ============================================================

CREATE TABLE orders (
  order_id BIGINT AUTO_INCREMENT PRIMARY KEY,
```

```
  order_no VARCHAR(30) NOT NULL UNIQUE,
  order_type ENUM('DINE_IN','TAKEAWAY') NOT NULL,
  table_id INT NULL,
  user_id INT NOT NULL,
  customer_name VARCHAR(80) NULL,
  order_datetime DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  status ENUM('DRAFT','PAID','CANCELLED') NOT NULL DEFAULT 'DRAFT',

  subtotal DECIMAL(12,2) NOT NULL DEFAULT 0 CHECK (subtotal >= 0),
  discount_amount DECIMAL(12,2) NOT NULL DEFAULT 0 CHECK (discount_amount
>= 0),
  tax_amount DECIMAL(12,2) NOT NULL DEFAULT 0 CHECK (tax_amount >= 0),
  total DECIMAL(12,2) NOT NULL DEFAULT 0 CHECK (total >= 0),

  note VARCHAR(255) NULL,

  CONSTRAINT fk_orders_table
    FOREIGN KEY (table_id) REFERENCES dining_tables(table_id)
    ON UPDATE CASCADE
    ON DELETE SET NULL,

  CONSTRAINT fk_orders_user
    FOREIGN KEY (user_id) REFERENCES users(user_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
```

```
) ENGINE=InnoDB;


CREATE INDEX idx_orders_datetime
ON orders(order_datetime);


CREATE INDEX idx_orders_user
ON orders(user_id);


CREATE TABLE order_items (
  order_item_id BIGINT AUTO_INCREMENT PRIMARY KEY,
  order_id BIGINT NOT NULL,
  product_id INT NOT NULL,
  qty INT NOT NULL CHECK (qty > 0),
  unit_price DECIMAL(12,2) NOT NULL CHECK (unit_price >= 0),
  line_total DECIMAL(12,2) NOT NULL CHECK (line_total >= 0),

  CONSTRAINT fk_items_order
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE,

  CONSTRAINT fk_items_product
    FOREIGN KEY (product_id) REFERENCES products(product_id)
    ON UPDATE CASCADE
    ON DELETE RESTRICT
```

```
) ENGINE=InnoDB;


CREATE INDEX idx_items_order
ON order_items(order_id);


CREATE INDEX idx_items_product
ON order_items(product_id);


CREATE TABLE payments (
  payment_id BIGINT AUTO_INCREMENT PRIMARY KEY,
  order_id BIGINT NOT NULL,
  method ENUM('CASH','QRIS','DEBIT','CREDIT') NOT NULL,
  paid_amount DECIMAL(12,2) NOT NULL CHECK (paid_amount >= 0),
  paid_at DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
  pay_status ENUM('PAID','REFUND') NOT NULL DEFAULT 'PAID',

  CONSTRAINT fk_payments_order
    FOREIGN KEY (order_id) REFERENCES orders(order_id)
    ON UPDATE CASCADE
    ON DELETE CASCADE
) ENGINE=InnoDB;


CREATE INDEX idx_payments_order
ON payments(order_id);
```

```
-- ============================================================

-- VIEW (DDL)

-- ============================================================


CREATE OR REPLACE VIEW v_invoice_detail AS

SELECT

  o.order_id,

  o.order_no,

  o.order_datetime,

  o.order_type,

  dt.table_no,

  u.full_name AS cashier,

  o.customer_name,

  p.product_name,

  c.category_name,

  oi.qty,

  oi.unit_price,

  oi.line_total,

  o.subtotal,

  o.discount_amount,

  o.tax_amount,

  o.total,

  o.status

FROM orders o

JOIN users u ON u.user_id = o.user_id
```

```
LEFT JOIN dining_tables dt ON dt.table_id = o.table_id

JOIN order_items oi ON oi.order_id = o.order_id

JOIN products p ON p.product_id = oi.product_id

JOIN categories c ON c.category_id = p.category_id;
```

Lampiran 2
Script SQL Data Manipulation Language (DML)

```
/* ============================================================

   DML – DATA MANIPULATION LANGUAGE

   ============================================================ */



-- ============================================================

-- 1. INSERT DATA MASTER

-- ============================================================



INSERT INTO roles (role_name)

VALUES ('admin'), ('kasir');



INSERT INTO users (role_id, full_name, username, password_hash)

VALUES

  (1, 'Admin Sistem', 'admin', 'hash_admin'),

  (2, 'Kasir 1', 'kasir1', 'hash_kasir1'),

  (2, 'Kasir 2', 'kasir2', 'hash_kasir2');



INSERT INTO dining_tables (table_no)
```

```
VALUES ('T1'), ('T2'), ('T3'), ('T4');


INSERT INTO categories (category_name)
VALUES ('Makanan'), ('Minuman'), ('Snack');


INSERT INTO products (category_id, product_name, unit, base_price, stock_qty)
VALUES
  (1, 'Nasi Goreng Spesial', 'porsi', 18000, 50),
  (1, 'Mie Goreng', 'porsi', 16000, 40),
  (1, 'Ayam Geprek', 'porsi', 20000, 35),
  (2, 'Es Teh', 'gelas', 5000, 100),
  (2, 'Es Jeruk', 'gelas', 7000, 80),
  (2, 'Kopi Susu', 'gelas', 12000, 60),
  (3, 'Kentang Goreng', 'porsi', 15000, 30),
  (3, 'Tahu Crispy', 'porsi', 10000, 45);


-- ============================================================
-- 2. SELECT (CEK DATA AWAL)
-- ============================================================


SELECT * FROM roles;
SELECT * FROM users;
SELECT * FROM categories;
SELECT * FROM products;
```

```
-- ============================================================

-- 3. INSERT TRANSAKSI (HASIL PROCEDURE)

-- ============================================================


-- Data ini dihasilkan setelah menjalankan:

-- CALL sp_create_order_demo(...)


SELECT * FROM orders;

SELECT * FROM order_items;


-- ============================================================

-- 4. INSERT & UPDATE PEMBAYARAN

-- ============================================================


INSERT INTO payments (order_id, method, paid_amount)

SELECT order_id, 'CASH', total

FROM orders

WHERE order_no = 'INV-0001';


UPDATE orders

SET status = 'PAID'

WHERE order_no = 'INV-0001';


-- ============================================================

-- 5. SELECT LAPORAN & CEK STOK
```

```
-- ============================================================


SELECT product_id, product_name, stock_qty

FROM products

ORDER BY product_id;


SELECT *

FROM v_invoice_detail

ORDER BY order_datetime;
```

Lampiran 3
Script SQL Transaction Control Language (TCL)

```
/* ============================================================

  TCL – TRANSACTION CONTROL LANGUAGE

  ============================================================ */


-- ============================================================

-- 1. TRANSAKSI PEMBUATAN ORDER (DALAM STORED PROCEDURE)

-- ============================================================


START TRANSACTION;


-- Pembuatan order (insert ke orders)

-- Pembuatan item order (insert ke order_items)
```

```
-- Perhitungan subtotal, diskon, pajak, dan total
-- Seluruh proses dijalankan secara atomik

COMMIT;

-- ============================================================
-- 2. EKSEKUSI TRANSAKSI MELALUI STORED PROCEDURE
-- ============================================================

CALL sp_create_order_demo(
  'INV-0001',
  'DINE_IN',
  1,
  2,
  'Budi',
  2000,
  0.10
);

CALL sp_create_order_demo(
  'INV-0002',
  'TAKEAWAY',
  NULL,
  3,
  'Sari',
```

```
  0,
  0.10
);

-- ============================================================
-- 3. TRANSAKSI PEMBAYARAN
-- ============================================================

START TRANSACTION;

INSERT INTO payments (order_id, method, paid_amount)
SELECT order_id, 'CASH', total
FROM orders
WHERE order_no = 'INV-0001';

UPDATE orders
SET status = 'PAID'
WHERE order_no = 'INV-0001';

COMMIT;
```

Lampiran 4
Script Query SQL JOIN dan Agregasi

```
/* ============================================================
  SCRIPT QUERY SQL
```

```
   JOIN, GROUP BY, HAVING, SUBQUERY
   ================================================================ */


-- ============================================================
-- 1. QUERY JOIN
-- Menampilkan detail nota lengkap berdasarkan nomor order
-- ============================================================

SELECT
  o.order_no,
  o.order_datetime,
  o.order_type,
  dt.table_no,
  u.full_name AS kasir,
  o.customer_name,
  p.product_name,
  c.category_name,
  oi.qty,
  oi.unit_price,
  oi.line_total,
  o.subtotal,
  o.discount_amount,
  o.tax_amount,
  o.total,
  o.status
```

```
FROM orders o
JOIN users u ON u.user_id = o.user_id
LEFT JOIN dining_tables dt ON dt.table_id = o.table_id
JOIN order_items oi ON oi.order_id = o.order_id
JOIN products p ON p.product_id = oi.product_id
JOIN categories c ON c.category_id = p.category_id
WHERE o.order_no = 'INV-0001'
ORDER BY p.product_name;


-- ============================================================
-- 2. QUERY AGREGASI (GROUP BY)
-- Total penjualan per hari
-- ============================================================

SELECT
  DATE(o.order_datetime) AS tanggal,
  COUNT(DISTINCT o.order_id) AS jumlah_nota,
  SUM(o.total) AS total_penjualan
FROM orders o
WHERE o.status <> 'CANCELLED'
GROUP BY DATE(o.order_datetime)
ORDER BY tanggal;


-- ============================================================
-- 3. GROUP BY + HAVING
```

```
-- Produk yang terjual lebih dari 2 item
-- ============================================================


SELECT
  p.product_name,
  SUM(oi.qty) AS total_qty_terjual
FROM order_items oi
JOIN orders o ON o.order_id = oi.order_id
JOIN products p ON p.product_id = oi.product_id
WHERE o.status <> 'CANCELLED'
GROUP BY p.product_name
HAVING SUM(oi.qty) > 2
ORDER BY total_qty_terjual DESC;


-- ============================================================
-- 4. SUBQUERY
-- Order dengan total di atas rata-rata
-- ============================================================


SELECT
  order_no,
  total
FROM orders
WHERE status <> 'CANCELLED'
  AND total > (
```

```
    SELECT AVG(total)
    FROM orders
    WHERE status <> 'CANCELLED'
  )
ORDER BY total DESC;
```

Lampiran 5
Repository GitHub

Link GitHub : https://github.com/Punipun85/UAS_BASIS_DATA

Struktur Folder GitHub :

Nota-Makanan-minuman

```
│
├── Dokumen/
│   └── laporan.docx
│   └── laporan.pdf
├── sql/
│   └── UAS_BASIS_DATA.sql
├──Poster/
│    └── Poster.pdf
└── README.md
```

DAFTAR PUSTAKA

1. Supriyanti, W. (2021). *Konsep Dasar Sistem Basis Data dengan MySQL*. Muhammadiyah University Press.
2. Suyanto, A. H. (2004). Basis Data Dan DBMS. *Yogyakarta: Universitas Gajah Mada*.
3. Hardini, M., Agarwal, V., Apriani, D., Widjaya, I. A., Setiawaty, E., & Nurasiah. (2025). *Application of database normalization in increasing data storage efficiency*. *International Transactions on Artificial Intelligence (ITALIC), 3*(2), 201-211. https://doi.org/10.33050/italic.v3i2.799
4. Arkan, M. N. (2025). *Perancangan Entity-Relationship Diagram (ERD) pada Basis Data Perpustakaan*. *Teknofile*. https://jurnal.nawansa.com/index.php/teknofile/article/view/411
5. Saha, B., Bagui, R., & Walsh-Earp, R. (2022). *Database design using entity-relationship diagrams: Essential to database design*. Routledge.
6. Hilman, N., Nafila, R., Putri, R. A., Zahwa, S., Nugroho, D. A., & Pujiono, I. P. (2025). Analisis Perbandingan Basis Data Relasional (SQL) dan Non-Relasional (NoSQL) Berdasarkan Efisiensi Penyimpanan dan Skalabilitas Data. *Jurnal RESTIKOM: Riset Teknik Informatika dan Komputer*, *7*(3), 367-373.
7. Yunianto, D. R., Putra, Y. E., & Rahmad, C. (2023). *Comparison of relational database modeling performance based on number of normalized entities*. *SMARTICS, 9*(1), 42-48. https://ejournal.unikama.ac.id/index.php/jst/article/view/8390
8. Budi, H. S. (2021). *Pengenalan Dasar SQL*. Deepublish.