Penerapan CRUD untuk Pengelolaan Data Transaksi pada Basis Data Penjualan

Disusun oleh:

Nama : Taufik Yoga Pratama
NPM : 5230411269
Kelas : VIII

PROGRAM STUDI INFORMATIKA

UNIVERSITAS TEKNOLOGI YOGYAKARTA

2023/2024

Kata Pengantar

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas segala limpahan rahmat dan karunia-Nya sehingga laporan ini dapat diselesaikan dengan baik dan tepat waktu. Laporan ini berjudul "Penerapan CRUD untuk Pengelolaan Data Transaksi pada Basis Data Penjualan" yang disusun sebagai bagian dari tugas praktikum mata kuliah Pemrograman Basis Data. Laporan ini berisi langkah-langkah pembuatan database, perancangan tabel, pembuatan relasi antar tabel, serta implementasi program CRUD (Create, Read, Update, Delete) menggunakan Python dan MySQL.

Kami menyadari bahwa dalam penyusunan laporan ini tidak terlepas dari kekurangan. Oleh karena itu, kami terbuka atas saran dan kritik yang membangun guna penyempurnaan laporan ini di masa mendatang. Semoga laporan ini dapat memberikan manfaat dan menjadi referensi yang berguna bagi pembaca dalam memahami pengelolaan data berbasis database.

Daftar Isi

BAB I

PENDAHULUAN

1.1. Latar Belakang

Dalam era digital yang serba otomatis, pengelolaan data menjadi salah satu aspek penting dalam pengembangan sistem informasi. Salah satu kebutuhan utama adalah menciptakan aplikasi yang dapat mengelola data secara efisien dan terstruktur. Sistem pengelolaan basis data (DBMS) menjadi komponen utama dalam hal ini, khususnya dalam kaitannya dengan pengelolaan transaksi pada berbagai sektor, seperti penjualan, perbankan, atau layanan publik.

Tugas ini berfokus pada implementasi konsep CRUD (Create, Read, Update, Delete) dalam sistem database berbasis MySQL menggunakan bahasa pemrograman Python. Pemilihan MySQL sebagai DBMS didasarkan pada keunggulannya dalam mendukung pengelolaan data skala kecil hingga besar. Sementara itu, Python dipilih karena kemudahan penggunaannya dan dukungan library yang beragam untuk integrasi database. Dengan menyusun sistem ini, diharapkan dapat memberikan pemahaman lebih dalam mengenai desain, implementasi, dan pengelolaan basis data secara praktis.

1.2. Rumusan Masalah

1. Bagaimana cara merancang database yang terdiri dari beberapa tabel dengan relasi yang sesuai?
2. Bagaimana cara mengimplementasikan CRUD untuk mengelola data transaksi menggunakan Python?
3. Bagaimana memastikan integritas data dalam database dengan menggunakan foreign key constraints?

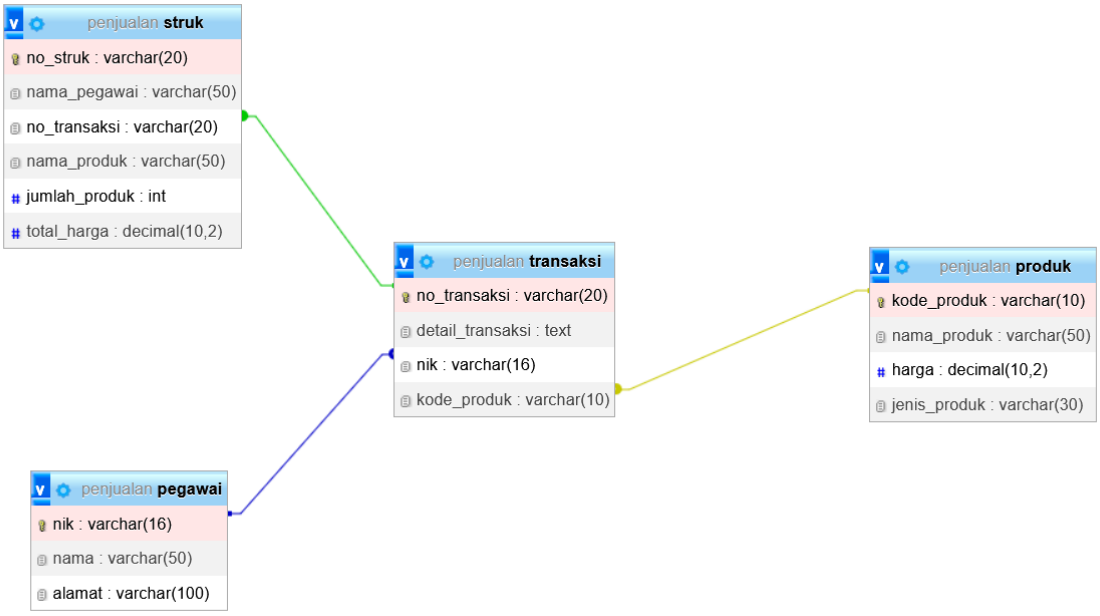
1.3. Tujuan

1. Memahami dan mengimplementasikan proses perancangan database relasional yang mencakup tabel dan relasinya.
2. Mengembangkan aplikasi sederhana berbasis Python untuk mengelola data transaksi dengan operasi CRUD.
3. Mengaplikasikan konsep integritas data pada database dengan relasi antar tabel menggunakan foreign key.

BAB II

PEMBAHASAN

2.1. **ERD**



2.2. **Langkah Langkah**

Langkah pertama dalam pembuatan sistem ini adalah merancang struktur database. Database diberi nama penjualan dan memiliki empat tabel utama, yaitu: transaksi, produk, pegawai, dan struk. Masing-masing tabel dirancang dengan atribut yang relevan, seperti tabel pegawai dengan atribut nik, nama, dan alamat, yang menggambarkan data karyawan. Relasi antar tabel dibuat untuk menjamin integritas data. Berikut adalah kode untuk membuat database dan tabel:

```
# Membuat database
mycursor.execute("CREATE DATABASE penjualan")

# Membuat tabel transaksi
mycursor.execute("""
CREATE TABLE transaksi (
    no_transaksi VARCHAR(20) NOT NULL PRIMARY KEY,
    detail_transaksi TEXT,
    nik VARCHAR(16),
    kode_produk VARCHAR(10)
)
""")

# Membuat tabel produk
mycursor.execute("""
```

```
CREATE TABLE produk (
    kode_produk VARCHAR(10) NOT NULL PRIMARY KEY,
    nama_produk VARCHAR(50),
    harga DECIMAL(10,2),
    jenis_produk VARCHAR(30)
)
""")

# Membuat tabel pegawai
mycursor.execute("""
CREATE TABLE pegawai (
    nik VARCHAR(16) NOT NULL PRIMARY KEY,
    nama VARCHAR(50),
    alamat VARCHAR(100)
)
""")

# Membuat tabel struk
mycursor.execute("""
CREATE TABLE struk (
    no_struk VARCHAR(20) NOT NULL PRIMARY KEY,
    nama_pegawai VARCHAR(50),
    no_transaksi VARCHAR(20),
    nama_produk VARCHAR(50),
    jumlah_produk INT,
    total_harga DECIMAL(10,2)
)
""")

# Membuat relasi antar tabel
mycursor.execute("ALTER TABLE transaksi ADD FOREIGN KEY (nik) REFERENCES
pegawai (nik)")
mycursor.execute("ALTER TABLE transaksi ADD FOREIGN KEY (kode_produk)
REFERENCES produk (kode_produk)")
mycursor.execute("ALTER TABLE struk ADD FOREIGN KEY (no_transaksi)
REFERENCES transaksi (no_transaksi)")
```

Langkah kedua adalah implementasi program CRUD. Program ini menggunakan bahasa Python dengan bantuan modul mysql.connector untuk menghubungkan ke database MySQL. Berikut adalah penjelasan fitur yang dikembangkan beserta baris kode terkait:

1. Create (Tambah Transaksi)

Pengguna dapat menambahkan data transaksi baru dengan memvalidasi nilai foreign key pada tabel pegawai dan produk. Berikut adalah kode untuk menambahkan transaksi:

```
def tambah_transaksi():
    nik = input("Masukkan NIK Pegawai: ")
    kode_produk = input("Masukkan Kode Produk: ")
    no_transaksi = input("Masukkan Nomor Transaksi: ")
    detail_transaksi = input("Masukkan Detail Transaksi: ")

    try:
        query = """
        INSERT INTO transaksi (nik, kode_produk, no_transaksi, detail_transaksi)
        VALUES (%s, %s, %s, %s)
        """
        cursor.execute(query, (nik, kode_produk, no_transaksi, detail_transaksi))
        db.commit()
        print("Transaksi berhasil ditambahkan!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

2. Read (Tampilkan Transaksi)

   Program menampilkan semua data transaksi yang ada di database menggunakan perintah SQL SELECT. Berikut adalah kode terkait:

```
def display_transaksi():
    try:
        query = "SELECT * FROM transaksi"
        cursor.execute(query)
        results = cursor.fetchall()
        if cursor.rowcount == 0:
            print("Tidak ada data transaksi.")
        else:
            print("Data Transaksi:")
            for row in results:
                print(row)
    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

3. Update (Perbarui Transaksi)

   Program memungkinkan pengguna memperbarui detail transaksi tertentu berdasarkan nomor transaksi. Berikut adalah kode untuk fitur ini:

```
def update_transaksi():
    no_transaksi = input("Masukkan Nomor Transaksi yang ingin diubah: ")
```

```
    detail_transaksi = input("Masukkan Detail Transaksi baru: ")

    try:
        query = "UPDATE transaksi SET detail_transaksi = %s WHERE no_transaksi
    = %s"
        cursor.execute(query, (detail_transaksi, no_transaksi))
        db.commit()
        if cursor.rowcount > 0:
            print("Transaksi berhasil diperbarui!")
        else:
            print("Nomor Transaksi tidak ditemukan.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

4. Delete (Hapus Transaksi)
   Pengguna dapat menghapus transaksi berdasarkan nomor transaksi. Berikut adalah kode untuk fitur ini:

```
def delete_transaksi():
    no_transaksi = input("Masukkan Nomor Transaksi yang ingin dihapus: ")

    try:
        query = "DELETE FROM transaksi WHERE no_transaksi = %s"
        cursor.execute(query, (no_transaksi,))
        db.commit()
        if cursor.rowcount > 0:
            print("Transaksi berhasil dihapus!")
        else:
            print("Nomor Transaksi tidak ditemukan.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")
```

2.3. **Salinan Codingan**

```
import mysql.connector

# ========== Step 1 membuat database ==========

# mydb = mysql.connector.connect(
#     host = "localhost",
#     user = "root",
#     password = ""
# )

# mycursor = mydb.cursor()
# mycursor.execute("CREATE DATABASE penjualan")
```

```
# ========== Step 2 membuat tabel tabel ==========

# mydb = mysql.connector.connect(
#     host = "localhost",
#     user = "root",
#     password = "",
#     database = "penjualan"
# )

# mycursor = mydb.cursor()

# Untuk tabel tarnsaksi
# mycursor.execute("CREATE TABLE transaksi (no_transaksi VARCHAR(20) NOT NULL
PRIMARY KEY, detail_transaksi TEXT, nik VARCHAR(16), kode_produk VARCHAR(10))")

# Untuk tabel prduk
# mycursor.execute("CREATE TABLE produk (kode_produk VARCHAR(10) NOT NULL PRIMARY
KEY, nama_produk VARCHAR(50), harga DECIMAL(10,2), jenis_produk VARCHAR(30))")

# Untuk tabel struk
# mycursor.execute("CREATE TABLE struk (no_struk VARCHAR(20) NOT NULL PRIMARY KEY,
nama_pegawai VARCHAR(50), no_transaksi VARCHAR(20), nama_produk VARCHAR(50),
jumlah_produk INT, total_harga DECIMAL(10,2))")

# Untuk tabel pegawai
# mycursor.execute("CREATE TABLE pegawai (nik VARCHAR(16) NOT NULL PRIMARY KEY,
nama VARCHAR(50), alamat VARCHAR(100))")

# ========== Step 3 membuat relasi antar tabel ==========

# Relasi tabel pegawai dengan tabel transaksi
# mycursor.execute("ALTER TABLE transaksi ADD FOREIGN KEY (nik) REFERENCES pegawai
(nik)")

# Relasi tabel struk dengan tabel transaksi
# mycursor.execute("ALTER TABLE struk ADD FOREIGN KEY (no_transaksi) REFERENCES
transaksi (no_transaksi)")

# Relasi tabel struk dengan tabel transaksi
# mycursor.execute("ALTER TABLE transaksi ADD FOREIGN KEY (kode_produk) REFERENCES
produk (kode_produk)")

# ========== Step 4 membuat program ==========

db = mysql.connector.connect(
  host="localhost",
  user="root",
  password="",
```

```
    database="penjualan"
)

cursor = db.cursor()

# Fungsi untuk menambahkan transaksi
def tambah_transaksi():
    # Note: sudah ada data dummy pada tabel pegawai, struk, dan produk
    nik = input("Masukkan NIK Pegawai: ")
    kode_produk = input("Masukkan Kode Produk: ")
    no_transaksi = input("Masukkan Nomor Transaksi: ")
    detail_transaksi = input("Masukkan Detail Transaksi: ")

    try:
        query = "INSERT INTO transaksi (nik, kode_produk, no_transaksi, detail_transaksi) VALUES
(%s, %s, %s, %s)"
        cursor.execute(query, (nik, kode_produk, no_transaksi, detail_transaksi))
        db.commit()
        print("Transaksi berhasil ditambahkan!")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

# Fungsi untuk membaca semua transaksi
def display_transaksi():
    try:
        query = "SELECT * FROM transaksi"
        cursor.execute(query)
        results = cursor.fetchall()
        if cursor.rowcount == 0:
            print("Tidak ada data transaksi.")
        else:
            print("Data Transaksi:")
            for row in results:
                print(row)
    except mysql.connector.Error as err:
        print(f"Error: {err}")

# Fungsi untuk memperbarui transaksi
def update_transaksi():
    no_transaksi = input("Masukkan Nomor Transaksi yang ingin diubah: ")
    detail_transaksi = input("Masukkan Detail Transaksi baru: ")

    try:
        query = "UPDATE transaksi SET detail_transaksi = %s WHERE no_transaksi = %s"
        cursor.execute(query, (detail_transaksi, no_transaksi))
        db.commit()
        if cursor.rowcount > 0:
            print("Transaksi berhasil diperbarui!")
```

```
        else:
            print("Nomor Transaksi tidak ditemukan.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

# Fungsi untuk menghapus transaksi
def delete_transaksi():
    no_transaksi = input("Masukkan Nomor Transaksi yang ingin dihapus: ")

    try:
        query = "DELETE FROM transaksi WHERE no_transaksi = %s"
        cursor.execute(query, (no_transaksi,))
        db.commit()
        if cursor.rowcount > 0:
            print("Transaksi berhasil dihapus!")
        else:
            print("Nomor Transaksi tidak ditemukan.")
    except mysql.connector.Error as err:
        print(f"Error: {err}")

while True:
    print("\nMenu CRUD Transaksi:")
    print("1. Tambah Transaksi")
    print("2. Lihat Semua Transaksi")
    print("3. Ubah Transaksi")
    print("4. Hapus Transaksi")
    print("5. Keluar")

    pilihan = input("Pilih menu (1-5): ")

    if pilihan == "1":
        # Note: sudah ada data dummy pada tabel pegawai, struk, dan produk
        tambah_transaksi()
    elif pilihan == "2":
        display_transaksi()
    elif pilihan == "3":
        update_transaksi()
    elif pilihan == "4":
        delete_transaksi()
    elif pilihan == "5":
        print("Keluar dari program.")
        break
    else:
        print("Pilihan tidak valid. Coba lagi.")
```

2.4. **Screenshot Program**

2.4.1. **Tambah**

```
Menu CRUD Transaksi:
1. Tambah Transaksi
2. Lihat Semua Transaksi
3. Ubah Transaksi
4. Hapus Transaksi
5. Keluar
Pilih menu (1-5): 1
Masukkan NIK Pegawai: 54321
Masukkan Kode Produk: 09876
Masukkan Nomor Transaksi: 12345
Masukkan Detail Transaksi: COD Cangkul
Transaksi berhasil ditambahkan!
```

2.4.2. **Lihat**

```
Menu CRUD Transaksi:
1. Tambah Transaksi
2. Lihat Semua Transaksi
3. Ubah Transaksi
4. Hapus Transaksi
5. Keluar
Pilih menu (1-5): 2
Data Transaksi:
('12345', 'COD Cangkul', '54321', '09876')

Menu CRUD Transaksi:
1. Tambah Transaksi
2. Lihat Semua Transaksi
3. Ubah Transaksi
4. Hapus Transaksi
5. Keluar
Pilih menu (1-5):
```

2.4.3. **Ubah**

```
Menu CRUD Transaksi:
1. Tambah Transaksi
2. Lihat Semua Transaksi
3. Ubah Transaksi
4. Hapus Transaksi
5. Keluar
Pilih menu (1-5): 3
Masukkan Nomor Transaksi yang ingin diubah: 12345
Masukkan Detail Transaksi baru: COD cangkul di titik te
ngah
Transaksi berhasil diperbarui!
```

2.4.4. **Hapus**

```
Menu CRUD Transaksi:
1. Tambah Transaksi
2. Lihat Semua Transaksi
3. Ubah Transaksi
4. Hapus Transaksi
5. Keluar
Pilih menu (1-5): 4
Masukkan Nomor Transaksi yang ingin dihapus: 12345
Transaksi berhasil dihapus!
```

BAB III

PENUTUP

3.1. Kesimpulan

Dari hasil implementasi dan pengujian program, dapat disimpulkan bahwa sistem CRUD berbasis Python dan MySQL ini mampu mengelola data transaksi secara efektif. Relasi antar tabel yang dirancang menggunakan foreign key berhasil memastikan integritas data di seluruh tabel database. Program yang dikembangkan juga memberikan pengalaman praktis dalam mengintegrasikan konsep basis data dengan pemrograman.

Penerapan CRUD ini tidak hanya memberikan solusi pengelolaan data, tetapi juga memperlihatkan pentingnya desain database yang baik untuk mendukung operasi yang andal. Dengan demikian, sistem ini dapat dijadikan sebagai dasar pengembangan aplikasi yang lebih kompleks di masa depan.