LAPORAN PROJECT UTS

PEMBUATAN SISTEM KASIR COFFEE SHOP

DISUSUN OLEH :

Defry Maulana

2110031802050

DOSEN PENGAMPU :

Dosen Pengampu :

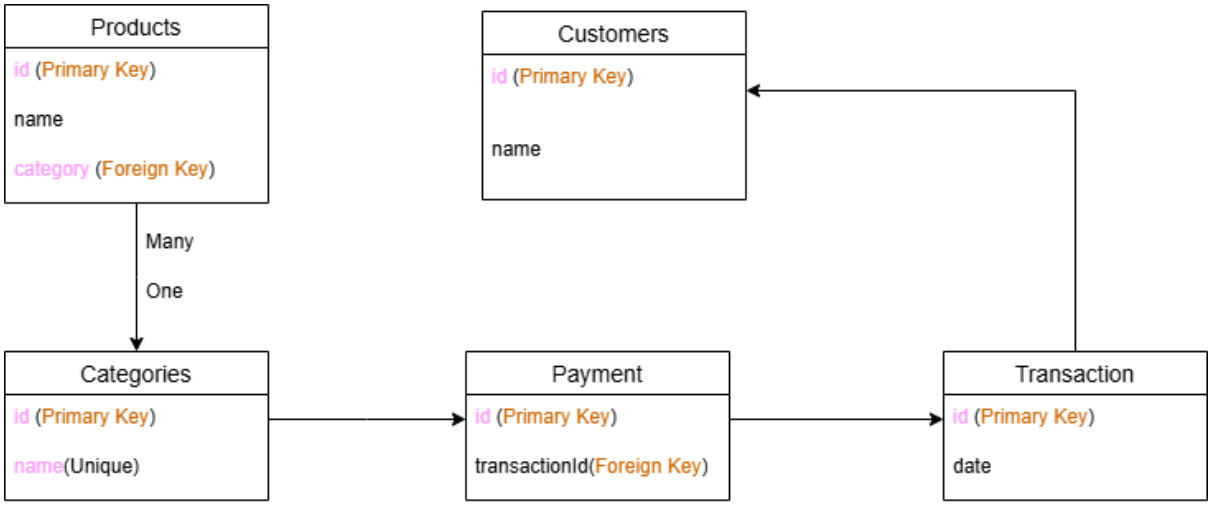
Torkis Nasution, M.Kom

PROGRAM STUDI TEKNIK INFORMATIKA

UNIVERSITAS SAINS DAN TEKNOLOGI INDONESIA

2024

1. Entity Relational Diagram



2. Penjelasan Entity Relational Diagram

Entities :

Products:

- id (PK): Primary Key, unik untuk setiap produk.
- name: Nama produk.
- categoryId (FK): Foreign Key yang merujuk ke Categories, menunjukkan kategori produk tersebut.

Categories:

- id (PK): Primary Key, unik untuk setiap kategori.
- name (Unique): Nama kategori yang harus unik.

Transactions:

- id (PK): Primary Key, unik untuk setiap transaksi.
- date: Tanggal transaksi dilakukan.

Customers:

- id (PK): Primary Key, unik untuk setiap pelanggan.
- name: Nama pelanggan.

Payments:

- id (PK): Primary Key, unik untuk setiap pembayaran.
- transactionId (FK): Foreign Key yang merujuk ke Transactions, menunjukkan transaksi yang terkait dengan pembayaran ini.

Hubungan

Products ke Categories:

- Setiap produk dapat dimiliki oleh satu kategori (One), tetapi satu kategori dapat memiliki banyak produk (Many). Ini adalah hubungan Many-to-One.

Payments ke Transactions:

- Setiap pembayaran terkait dengan satu transaksi (One), tetapi satu transaksi dapat memiliki banyak pembayaran (Many). Ini juga merupakan hubungan Many-to-One.

3. Physical Data Model

- **Tabel Products**

Kolom	Tipe Data	Keterangan
Id	INT (Primary Key)	Auto Increment
Name	Varchar(255)	Nama Produk
Price	Decimal (10,2)	Harga Produk
categoryId	INT (foreign Key)	Mengacu ke categories.id

- **Tabel Categories**

Kolom	Tipe Data	Keterangan
Id	INT (Primary Key	Auto Increment
Name	Varchar(255)	Nama Kategori (Unique)

- **Tabel Customers**

Kolom	Tipe Data	Keterangan
Id	INT (Primary Key)	Auto Increment
Name	Varchar(255)	Nama Pelanggan
Email	Varchar(255)	Email pelanggan

- **Tabel Transactions**

Kolom	Tipe Data	Keterangan
Id	INT (Primary Key)	Auto Increment
customerId	INT (Foreign Key)	Mengacu ke customers.id
productId	INT (Foreign Key)	Mengacu ke products.id
Quantity	INT	Jumlah Produk

- **Tabel Payments**

Kolom	Tipe data	Keterangan
Id	INT (Primary Key)	Auto Increment
transactionId	INT (Foreign Key)	Mengacu ke transactions.id
Amount	Decimal (10,2)	Total Pembayaran
Method	Varchar (255)	Metode Pembayaran

4. Penjelasan Physical Data Model

Tabel dan Perannya

1. **Products**:
   - Menyimpan informasi produk yang dijual di Coffee Shop.
   - Memiliki hubungan dengan tabel **Categories** untuk menentukan kategori dari setiap produk.
2. **Categories**:
   - Menyimpan daftar kategori produk, seperti *beverages*, *snacks*, atau *merchandise*.
   - Setiap kategori unik (tidak boleh ada nama kategori yang sama).
3. **Customers**:
   - Menyimpan data pelanggan yang melakukan transaksi.
   - Setiap pelanggan memiliki informasi unik, seperti nama dan email.
4. **Transactions**:
   - Menyimpan informasi setiap transaksi pembelian, termasuk produk yang dibeli, pelanggan yang membeli, dan jumlah produk.
   - Berfungsi sebagai penghubung antara pelanggan (**Customers**) dan produk (**Products**).
5. **Payments**:
   - Menyimpan data pembayaran untuk transaksi tertentu.
   - Setiap pembayaran terkait dengan satu transaksi saja (relasi *one-to-one*).

Hubungan Antar Tabel

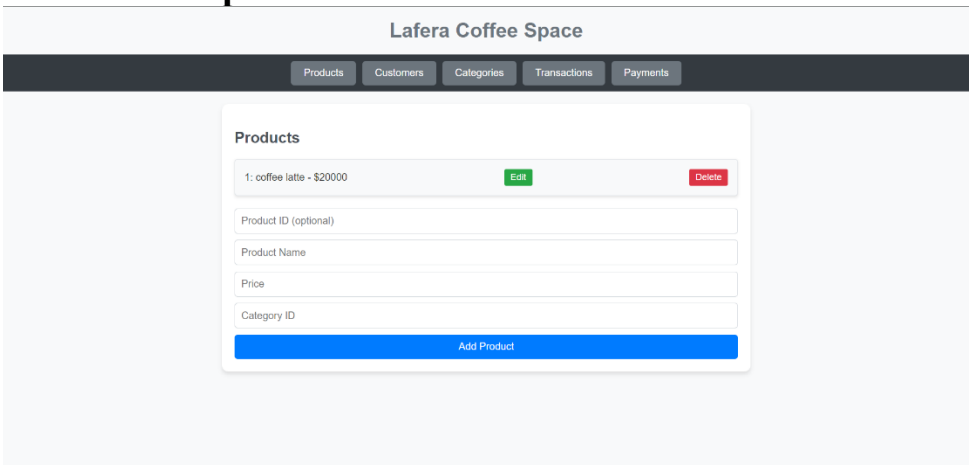
1. **Relasi Products ↔ Categories (Many-to-One)**:
   - Setiap produk memiliki satu kategori, tetapi satu kategori dapat memiliki banyak produk.
   - Foreign Key: Products.categoryId mengacu ke Categories.id.
2. **Relasi Transactions ↔ Customers (Many-to-One)**:
   - Setiap transaksi dilakukan oleh satu pelanggan, tetapi satu pelanggan dapat memiliki banyak transaksi.
   - Foreign Key: Transactions.customerId mengacu ke Customers.id.
3. **Relasi Transactions ↔ Products (Many-to-One)**:

- Setiap transaksi mencatat produk tertentu yang dibeli, tetapi satu produk dapat terlibat dalam banyak transaksi.
- Foreign Key: Transactions.productId mengacu ke Products.id.

4. **Relasi Payments ↔ Transactions (One-to-One)**:
   - Setiap pembayaran hanya terkait dengan satu transaksi, dan setiap transaksi hanya memiliki satu pembayaran.
   - Foreign Key: Payments.transactionId mengacu ke Transactions.id.

5. Program Menampilkan Data

- **Gambar Tampilan Hasil**



- **Penjelasan**

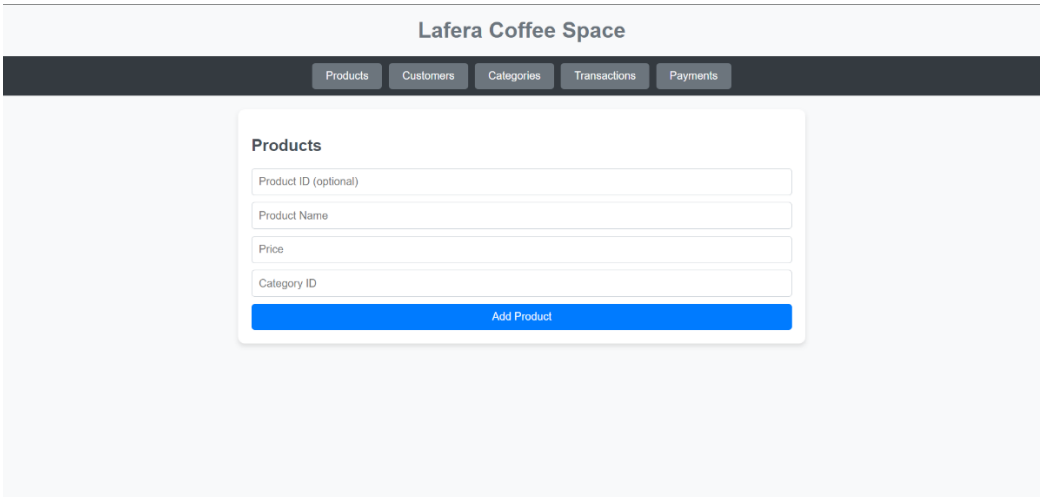
  Pada program ini,sistem dapat menginput data dan menampilkan daftar produk secara langsung.

- **Source Code**

```
function getProducts() {
    const transaction = db.transaction("products", "readonly");
    const store = transaction.objectStore("products");
    const request = store.getAll();

    request.onsuccess = function(event) {
        const productList = document.getElementById('product-list');
        productList.innerHTML = '';
        event.target.result.forEach(product => {
            const li = document.createElement('li');
            li.textContent = `${product.id}: ${product.name} - $${product.price}`;
            // Create edit button
        const editButton = document.createElement('button');
        editButton.textContent = 'Edit';
        editButton.onclick = () => editProduct(product.id);
        li.appendChild(editButton);

        // Create delete button
        const deleteButton = document.createElement('button');
        deleteButton.textContent = 'Delete';
        deleteButton.onclick = () => deleteProduct(product.id);
        li.appendChild(deleteButton);
            productList.appendChild(li);
        });
    };
}
```

6. Program Entri Data

- **Gambar Entri Data**



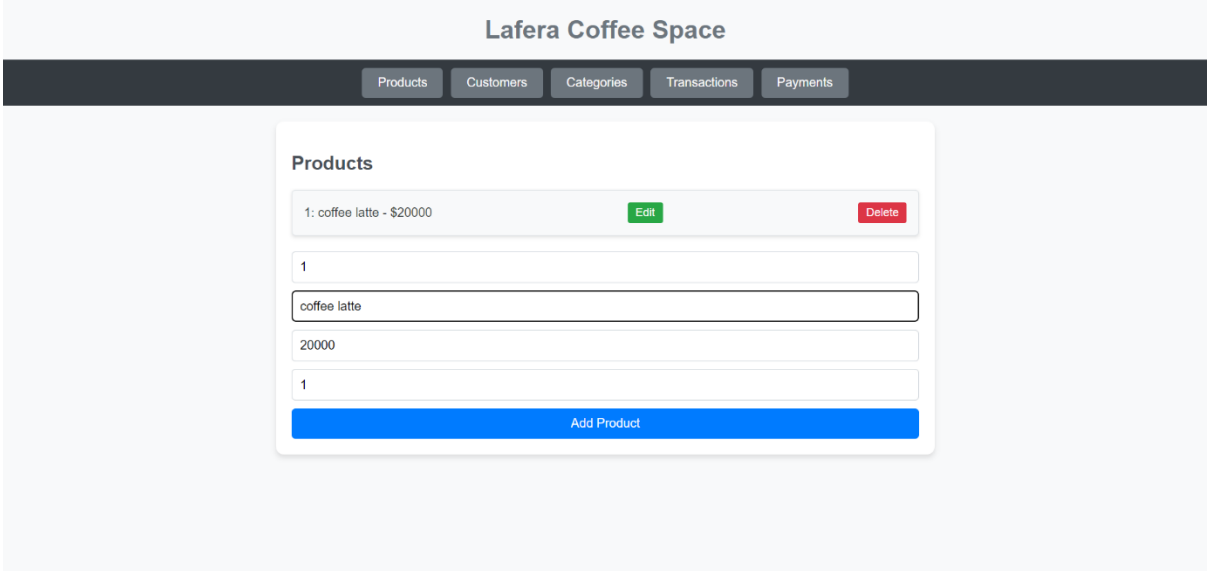
- **Penjelasan**
  Pada setiap tabel sistem ini,dapat melakukan entry data
- **Source Code**

```
document.getElementById('product-form').onsubmit = function(event) {
    event.preventDefault();
    const id = document.getElementById('product-id').value || Date.now();
    const name = document.getElementById('product-name').value;
    const price = parseFloat(document.getElementById('product-price').value);
    const categoryId = document.getElementById('product-categoryId').value;

    const transaction = db.transaction("products", "readwrite");
    const store = transaction.objectStore("products");
    store.put({ id, name, price, categoryId });

    transaction.oncomplete = function() {
        getProducts();
        document.getElementById('product-form').reset();
    };
};

function editProduct(productId) {
    const transaction = db.transaction("products", "readonly");
    const store = transaction.objectStore("products");
    const request = store.get(productId);

    request.onsuccess = function(event) {
        const product = event.target.result;
        if (product) {
            document.getElementById('product-id').value = product.id;
            document.getElementById('product-name').value = product.name;
            document.getElementById('product-price').value = product.price;
            document.getElementById('product-categoryId').value = product.categoryId;
        } else {
            console.error("Product not found");
        }
    };
```

```
    request.onerror = handleError;
}

document.getElementById('product-form').onsubmit = function(event) {
    event.preventDefault();
    const id = document.getElementById('product-id').value;
    const name = document.getElementById('product-name').value;
    const price = parseFloat(document.getElementById('product-price').value);
    const categoryId = document.getElementById('product-categoryId').value;

    const transaction = db.transaction("products", "readwrite");
    const store = transaction.objectStore("products");
    const request = store.put({ id, name, price, categoryId });

    request.onsuccess = function() {
        getProducts();
        document.getElementById('product-form').reset();
    };

    request.onerror = handleError;
};

function deleteProduct(productId) {
    const transaction = db.transaction("products", "readwrite");
    const store = transaction.objectStore("products");
    const request = store.delete(productId);

    request.onsuccess = function() {
        getProducts(); // Refresh the product list
    };

    request.onerror = handleError;
}
```

7. Program Update Data

- **Gambar Update Data**



- **Penjelasan**

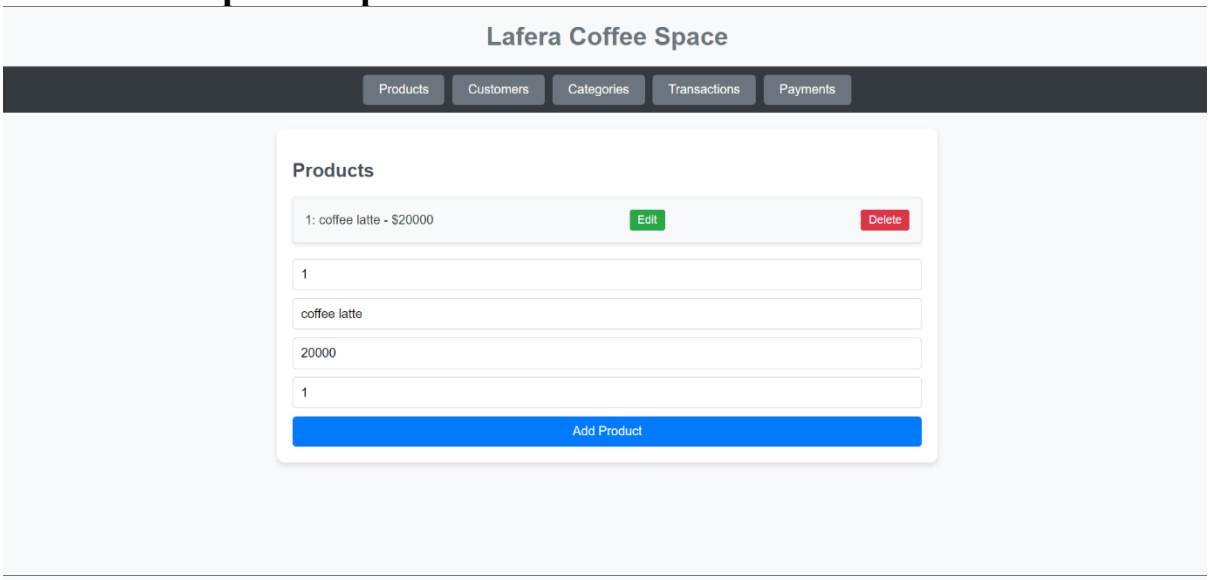
  Pada program update ini,sistem dapat melakukan edit pada setiap field

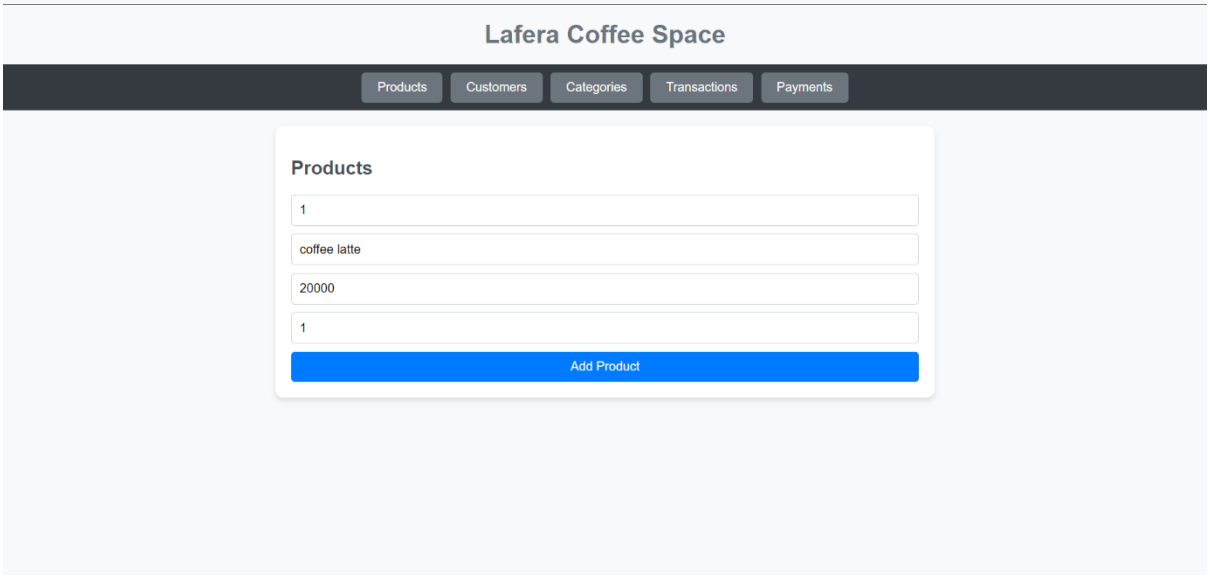
- **Source Code**

```
function editProduct(productId) {
    const transaction = db.transaction("products", "readonly");
    const store = transaction.objectStore("products");
    const request = store.get(productId);

    request.onsuccess = function(event) {
        const product = event.target.result;
        if (product) {
            document.getElementById('product-id').value = product.id;
            document.getElementById('product-name').value = product.name;
            document.getElementById('product-price').value = product.price;
            document.getElementById('product-categoryId').value = product.categoryId;
        } else {
            console.error("Product not found");
        }
    };

    request.onerror = handleError;
}
```

8. Program Hapus Data

- **Gambar Tampilan Hapus Data**





- **Penjelasan**

  Gambar 1. Menunjukkan bahwa masih ada daftar produk
  Gambar 2. Menunjukkan bahwa daftar produk telah dihapus

- **Source Code**

```
299  function deleteProduct(productId) {
300      const transaction = db.transaction("products", "readwrite");
301      const store = transaction.objectStore("products");
302      const request = store.delete(productId);
303
304      request.onsuccess = function() {
305          getProducts(); // Refresh the product list
306      };
307
308      request.onerror = handleError;
309  }
310
```

9. Penutup

- **Kesimpulan**

  Proyek pembuatan sistem kasir untuk Coffee Shop ini telah dirancang dan diimplementasikan dengan pendekatan yang sistematis. Sistem ini mampu menangani proses entri data, pembaruan, penghapusan, serta pengelolaan transaksi secara efisien. Dengan desain database yang baik, sistem ini dapat mendukung operasional Coffee Shop dan memberikan kemudahan dalam pengelolaan data produk, pelanggan, transaksi, serta pembayaran.Sistem yang dikembangkan diharapkan mampu meningkatkan produktivitas, mengurangi kesalahan manusia, dan memberikan pengalaman yang lebih baik kepada pelanggan.

- **Saran**

1. **Pengembangan Fitur Tambahan**
   Untuk meningkatkan kapabilitas sistem, dapat ditambahkan fitur seperti laporan penjualan otomatis, integrasi dengan pembayaran digital, atau manajemen stok secara real-time.
2. **Peningkatan Keamanan Sistem**
   Mengimplementasikan enkripsi pada data pelanggan dan transaksi untuk memastikan keamanan informasi.
3. **Uji Coba pada Skala Nyata**
   Sebaiknya sistem diuji pada lingkungan operasional Coffee Shop untuk mengidentifikasi potensi perbaikan sebelum diterapkan sepenuhnya.
4. **Pelatihan Pengguna**
   Menyediakan pelatihan kepada staf Coffee Shop agar dapat menggunakan sistem ini secara efektif.

Dengan adanya saran ini, diharapkan sistem dapat terus dikembangkan dan memberikan manfaat yang optimal bagi penggunanya.