LAUNDRY SHOP

Oracle SQL Database Project

CSE 3110
Database Systems Laboratory

Name: Tirtho Mondal
Department: Computer Science and Engineering
Roll: 2007117
Year: 3rd
Term: 1st

Introduction

In the contemporary business landscape, efficient data management is pivotal for operational success. This holds particularly true for businesses like laundry shops, where a multitude of transactions occur daily. This document presents an in-depth analysis of the design and implementation of a relational database tailored for a laundry shop. By leveraging structured data storage and retrieval mechanisms, the laundry shop can streamline its operations, enhance customer service, and drive profitability.

Project Overview:

The project aims to develop a robust relational database system tailored specifically for managing the operations of a laundry shop. It encompasses the creation of tables to store essential data such as cloth details, client information, employee records, order details, and payment transactions. By organizing data in a structured format, the database facilitates seamless access, manipulation, and analysis of information critical for day-to-day operations.

Importance of the Database in Laundry Shop

A well-designed database system is indispensable for the smooth functioning of a laundry shop. It serves as the backbone for organizing, storing, and retrieving critical information related to inventory, orders, clients, and payments. With a centralized database, the laundry shop can efficiently track inventory levels, manage customer orders, assign tasks to employees, and process payments in a timely manner. This not only streamlines operations but also enhances decision-making capabilities and enables the business to deliver high-quality services consistently.

Project Objectives

In this project, the primary aim is to design a robust relational database schema tailored specifically for managing data related to laundry services. By creating a well-structured database, we intend to streamline the management of various aspects such as cloth types, client information, employee details, order details, and payment records. This database will serve as a centralized repository for all laundry-related data, facilitating efficient data organization and retrieval.

Designing a Relational Database Schema for Managing Laundry-Related Data:

- Develop a structured database schema capable of efficiently storing and organizing data related to laundry services, including information about cloths, clients, employees, orders, and payments.
- Ensure data integrity and consistency by implementing appropriate constraints, such as primary keys, foreign keys, and data types, to enforce relational integrity and prevent inconsistencies.
- Design the schema to accommodate future scalability and flexibility requirements, allowing for easy expansion and modification as the business grows or evolves.

Implementing SQL Queries for Retrieving, Updating, and Managing Data within the Database:

- Develop a set of SQL queries for performing various operations on the database, including retrieving information, updating records, inserting new data, and deleting obsolete or erroneous entries.
- Create queries to support essential functionalities such as adding new clients, processing orders, managing inventory, tracking payments, and generating reports for business analysis and decision-making.
- Optimize query performance through indexing, query optimization techniques, and efficient use of SQL features to ensure fast and responsive data access, especially for frequently executed operations.

Database Design & Overview

The database schema of Laundry Shop Management are consist of the following tables:

- Cloths: Stores information about different types of cloths including their type, color, size, and fabric type.
- Clients: Contains details about clients such as their names, contact information, and the cloth they have assigned.
- Employees: Stores information about employees including their names, contact details, and positions within the laundry shop.
- OrderDetails: Tracks the details of each order including the cloth ordered, client placing the order, employee handling the order, quantity, unit price, and subtotal.
- Payment: Stores payment details including payment ID, associated order ID, payment date, and amount paid.

Rationale Behind the Design Decisions

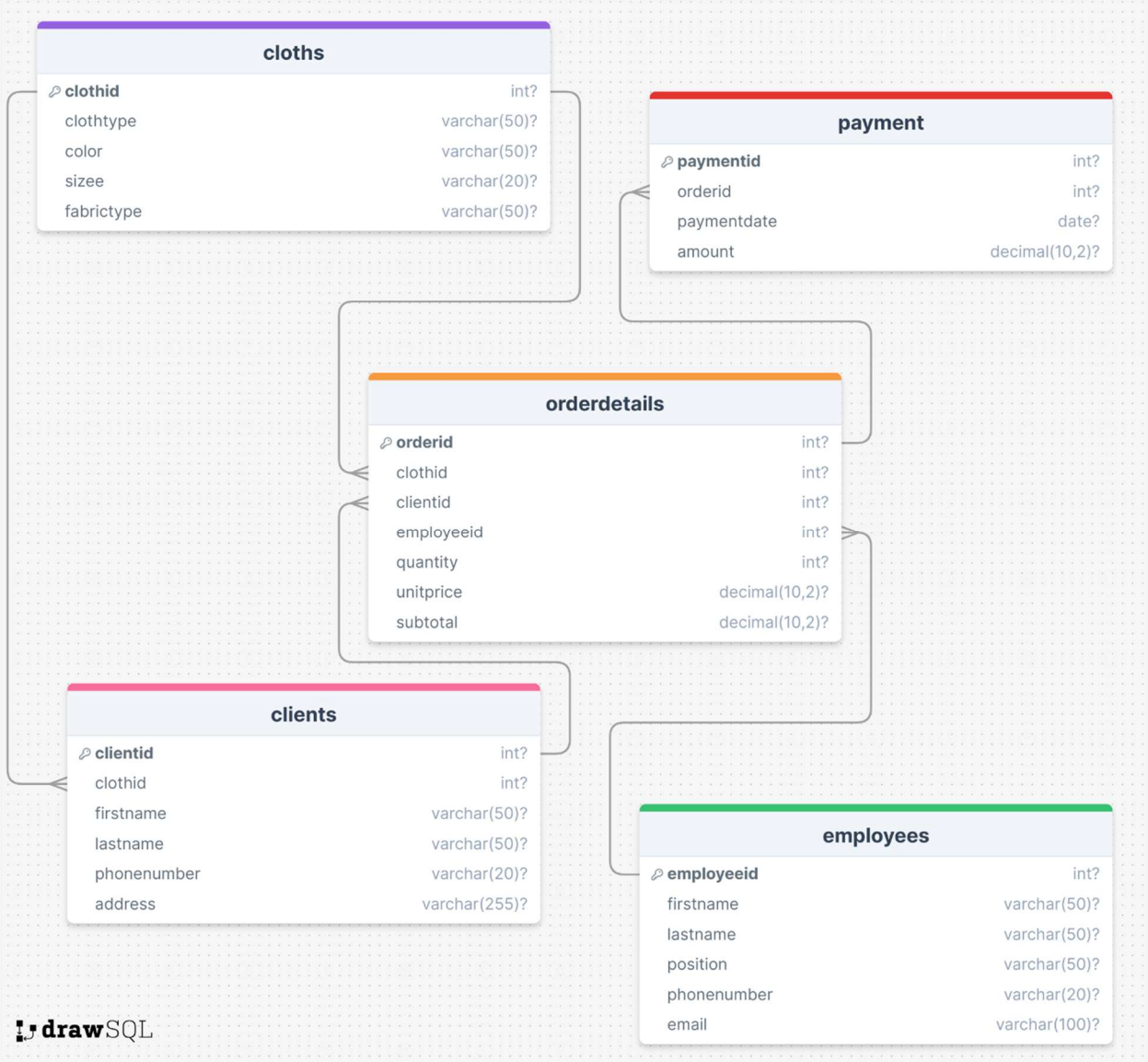
The database schema is designed to capture all essential aspects of laundry management including inventory, orders, clients, employees, and payments. Each table is structured to minimize redundancy and maintain data integrity, ensuring accurate and reliable information retrieval.

Table Design

The tables are interconnected through foreign key constraints, establishing relationships that facilitate data retrieval and integrity maintenance.
The tables are given bellow

Table	SQL
Cloths	create table cloths (clothid int primary key, clothtype varchar(50), color varchar(50), sizee varchar(20), fabrictype varchar(50));
Clients	create table clients (clientid int primary key, clothid int, firstname varchar(50), lastname varchar(50), phonenumber varchar(20), address varchar(255), foreign key (clothid) references cloths(clothid));
Employees	create table employees (employeeid int primary key, firstname varchar(50), lastname varchar(50), position varchar(50), phonenumber varchar(20), email varchar(100));
OderDetails	create table orderdetails (orderid int primary key, clothid int, clientid int, employeeid int, quantity int, unitprice decimal(10, 2), subtotal decimal(10, 2), foreign key (clothid) references cloths(clothid), foreign key (clientid) references clients(clientid), foreign key (employeeid) references employees(employeeid));
Payment	create table payment (paymentid int primary key, orderid int, paymentdate date, amount decimal(10, 2), foreign key (orderid) references orderdetails(orderid));

Entity-Relationship Diagram (E-R Diagram)



Normalization:

Normalization was employed to eliminate data redundancy and dependency anomalies, thereby enhancing data integrity and minimizing storage space requirements. The tables are normalized up to the third normal form (3NF), ensuring efficient data management and reducing the risk of inconsistencies

SQL Queries and Functionality

SQL queries and functionality are indispensable components of our project, empowering users to seamlessly engage with the database. Here, we outline pivotal SQL queries and functionalities crucial for optimal system operation.

❖ Data Retrieval:

- Retrieve Clients Information:

```
select * from clients where address = 'Khulna, Bangladesh';
```

This SQL query retrieves all columns from the clients table where the address column matches the value 'Khulna, Bangladesh'.

- The Sum of the subtotal of a clients:

```
select sum(subtotal) as total_subtotal from orderdetails where
clientid = 'x';
```

This SQL query calculates the sum of the subtotal column (renamed as total_subtotal) from the orderdetails table where the clientid matches a specified value ('x').

- The information of client who paid the highest amount single transection(using Nested query)

```
select * from clients where clientid = (
select clientid from orderdetails where unitprice = (
select max(unitprice) from orderdetails));
```

This SQL query selects all columns from the clients table where the clientid matches the result of a nested subquery. The nested subquery finds the clientid from the orderdetails table where the unitprice equals the maximum unitprice from the orderdetails table.

❖ Add column

- Age column to the employees table

```
alter table employees add age number;
```

❖ Searching

- Find the clients who's firstname started with 'T' and lastname with 'M'

```
select * from clients where firstname like 'T%' and lastname like
'M%';
```

- Find the average age of employees grouped by their position, where the average age is greater than 32

```
select position, avg(age) as avg_age from employees group by position
having avg(age) > 32;
```

This SQL query selects the position column and calculates the average age (avg_age) of employees for each distinct position from the employees table. Then, it filters the results to only include positions where the average age is greater than 32.

❖ Update:

- The information of the cloths

```
update cloths set clothtype = 'shirt', color = 'black', sizee = 'xl'
where clothid = 3;
```

❖ Advanced SQL Command;

- The eldest employee

```
set serveroutput on
create or replace function find_eldest_employee return varchar2 is
    eldest_employee_name varchar2(100);
begin
    select firstname || ' ' || lastname into eldest_employee_name
    from employees
    where age = (select max(age) from employees);

    return eldest_employee_name;
end;
/
declare
    eldest_employee varchar2(100);
begin
    eldest_employee := find_eldest_employee();
    dbms_output.put_line('The eldest employee is: ' ||
eldest_employee);
end;
/
```

This PL/SQL code defines a function named find_eldest_employee that returns the full name of the eldest employee based on their age. Within a DECLARE block, it calls this function and assigns the result to a variable, then prints out a message indicating the name of the eldest employee using DBMS_OUTPUT.PUT_LINE. The function queries the employees table to find the employee with the maximum age and concatenates their first and last names to form the full name returned by the function.

- Clots Information :

```
set serveroutput on
declare
    cursor cloth_cursor is select * from cloths;
    cloth_row cloths%rowtype;
    row_counter number := 0;
begin
    open cloth_cursor;
    fetch cloth_cursor into cloth_row.clothid, cloth_row.clothtype,
cloth_row.color, cloth_row.sizee, cloth_row.fabrictype;
    while cloth_cursor%found loop
        row_counter := row_counter + 1;
        dbms_output.put_line('Cloth ID: ' || cloth_row.clothid || '
Cloth Type: ' || cloth_row.clothtype || ' Color: ' || cloth_row.color
|| ' Size: ' || cloth_row.sizee || ' Fabric Type: ' ||
cloth_row.fabrictype);
        dbms_output.put_line('Row count: ' || row_counter);
        fetch cloth_cursor into cloth_row.clothid,
cloth_row.clothtype, cloth_row.color, cloth_row.sizee,
cloth_row.fabrictype;
    end loop;
    close cloth_cursor;
end;
/
```

This PL/SQL block retrieves all rows from the cloths table using a cursor. It then iterates over each row, fetching the values into variables of type cloths%rowtype. Inside the loop, it prints out the details of each cloth, including its ID, type, color, size, and fabric type, along with a row count. Finally, it closes the cursor. This block effectively outputs the details of all cloths stored in the database.

➢ Update the order subtotal:

```
create or replace trigger update_order_subtotal
before insert or update of quantity, unitprice on orderdetails
for each row
begin
    :NEW.subtotal := :NEW.quantity * :NEW.unitprice;
    update payment set amount = :NEW.subtotal WHERE orderid =
:NEW.orderid;
end;
/
```

This trigger, named update_order_subtotal, is designed to automatically update the subtotal column in the orderdetails table whenever a new order is inserted or the quantity or unitprice of an existing order is updated. It calculates the subtotal by multiplying the quantity and unit price of the order. Additionally, it updates the corresponding payment amount in the payment table to reflect the new subtotal for the order identified by the ordered

✓ **Advantages:**

- ➢ Efficient data management facilitates organized storage and retrieval of critical information.
- ➢ Improved customer service results from centralized data management, enhancing order processing and inventory tracking.
- ➢ Enhanced operational efficiency is achieved through automation of processes like updating order subtotals.
- ➢ Scalability and flexibility are ensured with a designed schema allowing for easy expansion and modification.
- ➢ Data integrity and consistency are maintained through normalization and appropriate constraints.

✓ Disadvantages:

- Complexity of implementation requires careful planning and execution.
- Cost and resource intensiveness entail financial investment and skilled personnel.
- Potential for technical issues such as database errors or performance bottlenecks exists.
- Dependency on technology infrastructure may lead to disruptions or failures impacting operations.
- Training and adoption challenges may arise for employees unfamiliar with the system.

Conclusion:

In conclusion, the development and implementation of a relational database tailored for a laundry shop offer numerous benefits in terms of operational efficiency, data accuracy, and decision support. By centralizing data management processes, the database enables the laundry shop to optimize its operations, enhance customer satisfaction, and drive business growth. Moving forward, regular maintenance and updates will be essential to ensure the database remains aligned with evolving business needs and technological advancements.

References:

- https://www.academia.edu/41847521/Laundry_Management_System_Design_and_Implementation
- https://itsourcecode.com/free-projects/database-design-projects/laundry-management-system-database-design/
- https://capstoneguide.com/laundry-management-system-capstone-project-document/
- https://drawsql.app/