Customer Ticket Management System - Database Design

Sandeep Batta
sabatta@iu.edu
Indiana University

Satya Priyanka Ponduru
sponduru@iu.edu
Indiana University

Vikranth Nallapuneni
vnallap@iu.edu
Indiana University

We've established a **MySQL** instance utilizing the "**Cloud SQL**" service on Google Cloud Platform. The raw dataset obtained from Kaggle was initially in 1st Normal Form, requiring preprocessing before proceeding with data modelling.

Since the original dataset featured a unique user for every ticket, we opted to sample 500 users and randomly assign these individuals to all tickets. Following this preprocessing, inserted data into MySQL table.

Database Schema

The dataset contains a variety of information about customers and their interactions with products and support services. It includes details like customers' names, email addresses, ages, and genders. It also records the products they've purchased and when they made those purchases. Additionally, the dataset covers details about support tickets, like their types, subjects, descriptions, statuses, resolutions, priorities, and channels of communication.

Normalization Process:

The Normalization Process is organizing and structuring a relational database to minimize redundancy and dependency, ensuring data integrity and efficiency.

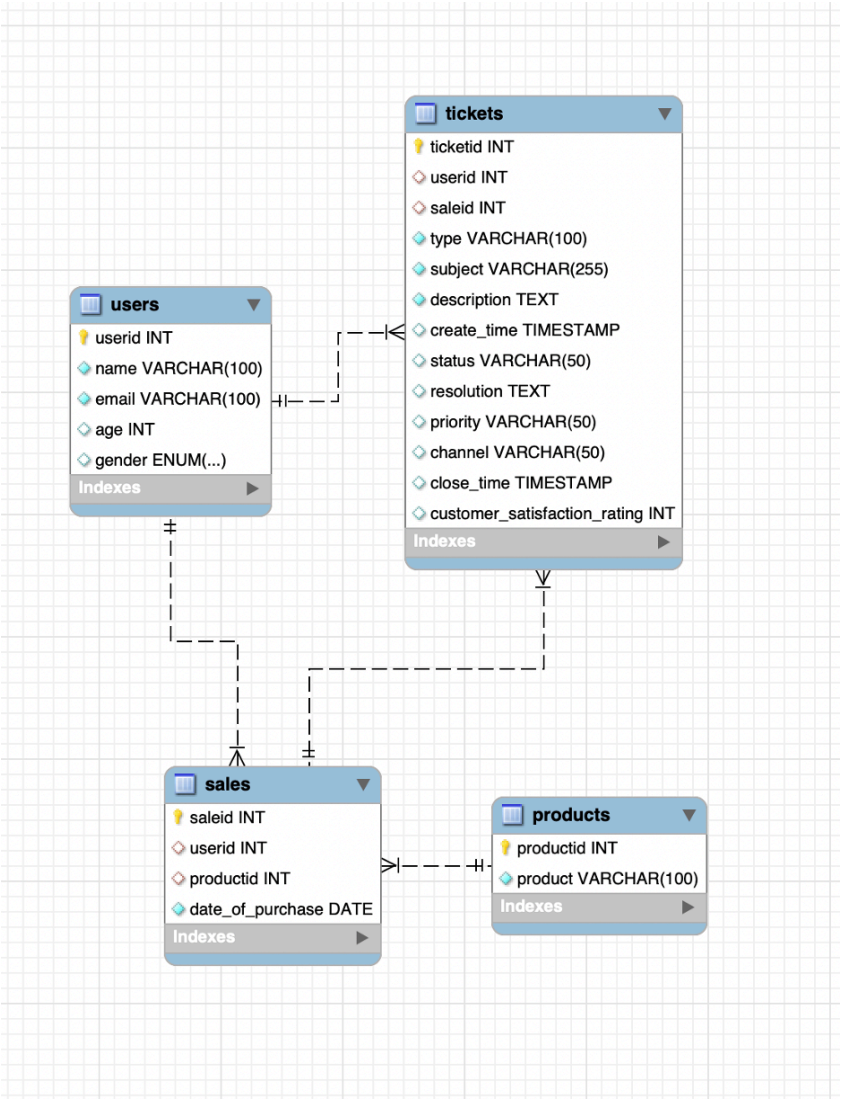
- **First Normal Form(1NF) :** In this we will be ensuring that each column holds only a single value. For example, each ticket in the Tickets table has a single TicketID, and each sale in the Sales table has a single SaleID.

- **Second Normal Form (2NF)** : In this for partial dependencies should be removed. It can be achieved by breaking down tables into smaller, more granular tables, ensuring that each table represents a single entity and has no partial dependencies. For example, the Sales table has been created to link users to products, removing any partial dependencies from the Tickets table.

- **Third Normal Form (3NF)** : In this we will remove Transitive Dependencies. It can be achieved by further breaking down tables to eliminate any transitive dependencies and ensure that each attribute in a table is directly dependent on the primary key. For

example, the Tickets table contains no transitive dependencies as each attribute is directly dependent on the TicketID (primary key).

- We have divided the original data set with 17 columns into 4 different tables.



Tables

1. Product Table:
    - It has 2 columns ProductID and Product
    - This table stores information about the products available for purchase
    - Product column holds the name or description of the product.
2. Sales Table:
    - It has 4 columns SalesID, ProductID, UserID and Date of Purchase
    - Records sales transactions, providing details on which products were sold to which users and when
    - Date of Purchase records the timestamp of each transaction
3. Tickets Table:

- Columns are TicketID, UserID, SaleID, Type, Subject, Description, Create_Time, Status, Resolution, Priority, Channel, Close_Time and Customer_Satisfaction_Rating
- Manages support tickets raised by users, providing detailed information about each ticket and its resolution

4. Users Table:
    - Columns are UserID, Name, Email, Age and Gender
    - It Stores information about users, including their name, email, age, and gender
    - Email serves as a unique identifier for user login and communication purposes
    - Age and Gender provide demographic information about users

Database Constraints

The database constraints specified in the "Customer Ticket Management System - Database Design" are as follows:

1. **Non-Null Constraints**:
    - In the `users` table, the `name`, `email`, and optionally `age` and `gender` fields must not be null, ensuring that every user record has a name and email.
    - In the `products` table, the `product` field must not be null, ensuring that every product record has a product name or description.
    - In the `sales` table, the `date_of_purchase` must not be null, ensuring every sales transaction record includes the date the purchase was made.
    - In the `tickets` table, the `type`, `subject`, and `description` fields must not be null, ensuring every ticket has these essential details for processing.
2. **Uniqueness Constraints**:
    - The `email` field in the `users` table has a uniqueness constraint, ensuring that each user's email is unique across the database.
3. **Primary Key Constraints**:
    - Each table (`users`, `products`, `sales`, `tickets`) has a primary key (`userid`, `productid`, `saleid`, `ticketid` respectively), auto-incremented, ensuring a unique identifier for each record.
4. **Foreign Key Constraints**:
    - The `sales` table references `users(userid)` and `products(productid)` as foreign keys, linking sales records to specific users and products.
    - The `tickets` table references `users(userid)` and `sales(saleid)` as foreign keys, linking ticket records to specific users and sales transactions.
5. **Data Type Constraints**:
    - The `gender` field in the `users` table uses an ENUM data type with values ('Male', 'Female', 'Other'), restricting the input to these specified values.
6. **Default Values**:

- In the `tickets` table, `create_time` has a default value of the current timestamp, `status` defaults to 'Open', `priority` defaults to 'Low', ensuring that new tickets have initial statuses and priorities unless specified otherwise.

Code - MySQL

We've created a comprehensive SQL script to set up a database for our Customer Ticket Management System (CTMS) project. This script establishes tables for managing users, products, sales transactions, and support tickets, while also populating them with data from an existing source.

Below are common SQL queries we will use in the application.

Get all closed tickets

```
-- Vikranth Nallapuneni
SELECT u.name AS username, p.product AS product_purchased, t.*
FROM tickets t
JOIN users u ON t.userid = u.userid
JOIN sales s ON t.saleid = s.saleid
JOIN products p ON s.productid = p.productid
WHERE t.status = 'Open' AND t.priority = 'Critical';
```

Get open critical tickets

```
-- Sandeep Batta
SELECT u.name AS username, p.product AS product_purchased, t.*
FROM tickets t
JOIN users u ON t.userid = u.userid
JOIN sales s ON t.saleid = s.saleid
JOIN products p ON s.productid = p.productid
WHERE t.status = 'Open' AND t.priority = 'Critical';
```

Get open tickets of a user

```
-- SATYA PRIYANKA PONDURU
SELECT u.name AS username, p.product AS product_purchased, t.*
FROM tickets t
JOIN users u ON t.userid = u.userid
JOIN sales s ON t.saleid = s.saleid
JOIN products p ON s.productid = p.productid
WHERE t.status = 'Open'
AND u.name = 'Grace Carroll';
```

Contribution

Name	Task	Contribution	Avg Time Spent
Sandeep Batta	Preprocessing and MySQL instance	Raw dataset preprocessing, instance creation and management on GCP	5 Hours
	SQL Code	SQL code outline, created and populated database, raw data table, Tickets table	
	Documentation	Created the outline, introduction and code section	
Satya Priyanka Ponduru	Normalization	Researched about Normalization process and data modelling	5 hours
	SQL Code	Users and Products table design and Insertion of data into it	
	Documentation	Normalization Process Tables	
Vikranth Nallapuneni	Database Constraints	Created various database constraints suitable for the schema of each table	5 hours
	SQL Code	Wrote SQL Code for sales table by joining various tables and retrieving required attributes	
	Documentation	Database Constraints	