A Project Report On

Banking Database System

Summited by,

Abhishek Tarpara

(180470107056)

Under Guidance of

Prof. Tejas Pataliya

(HOD of Computer science)

Academic year (2019-20)

राष्ट्राय स्वाहा इदं न मम।

VVP Engineering College Rajkot

1.Introduction:

The purpose of this report is to provide an overview of the banking database project. The project aims to design and implement a robust and efficient database system tailored for managing banking operations. By leveraging a well-designed database schema, this project intends to streamline account management, transaction processing, and customer information storage in a secure and organized manner.

The primary objectives of the banking database project are to enhance operational efficiency, improve data integrity, and enable comprehensive reporting and analysis capabilities. By developing a well-structured database, we aim to automate various banking processes, reduce manual errors, and provide a reliable foundation for data-driven decision-making within the banking institution.

This report will delve into the database design, data population process, and the functionality offered by the database system. It will explore the relationships between entities, discuss the implemented constraints and security measures, and showcase the capabilities of the database through sample queries and reports.

Furthermore, this report will address the reporting and analysis requirements of the project. It will highlight key performance indicators (KPIs) relevant to the banking domain and demonstrate how the database system can generate insightful reports and provide valuable business intelligence.

While the project has its defined scope, it is essential to acknowledge the limitations and constraints that may impact its implementation. These limitations will be discussed to provide a comprehensive understanding of the project boundaries and potential areas for future enhancements.

By the end of this report, readers will gain insights into the design, functionality, and potential benefits of the banking database project. The report will serve as a valuable resource for project stakeholders, system administrators, and decision-makers involved in the banking institution.

2. ER model:

Database design for a banking system involves creating a structured and efficient representation of the system's data and its relationships. One popular approach to designing a database is using the Entity-Relationship (ER) model, which helps visualize entities, attributes, and the relationships between them.

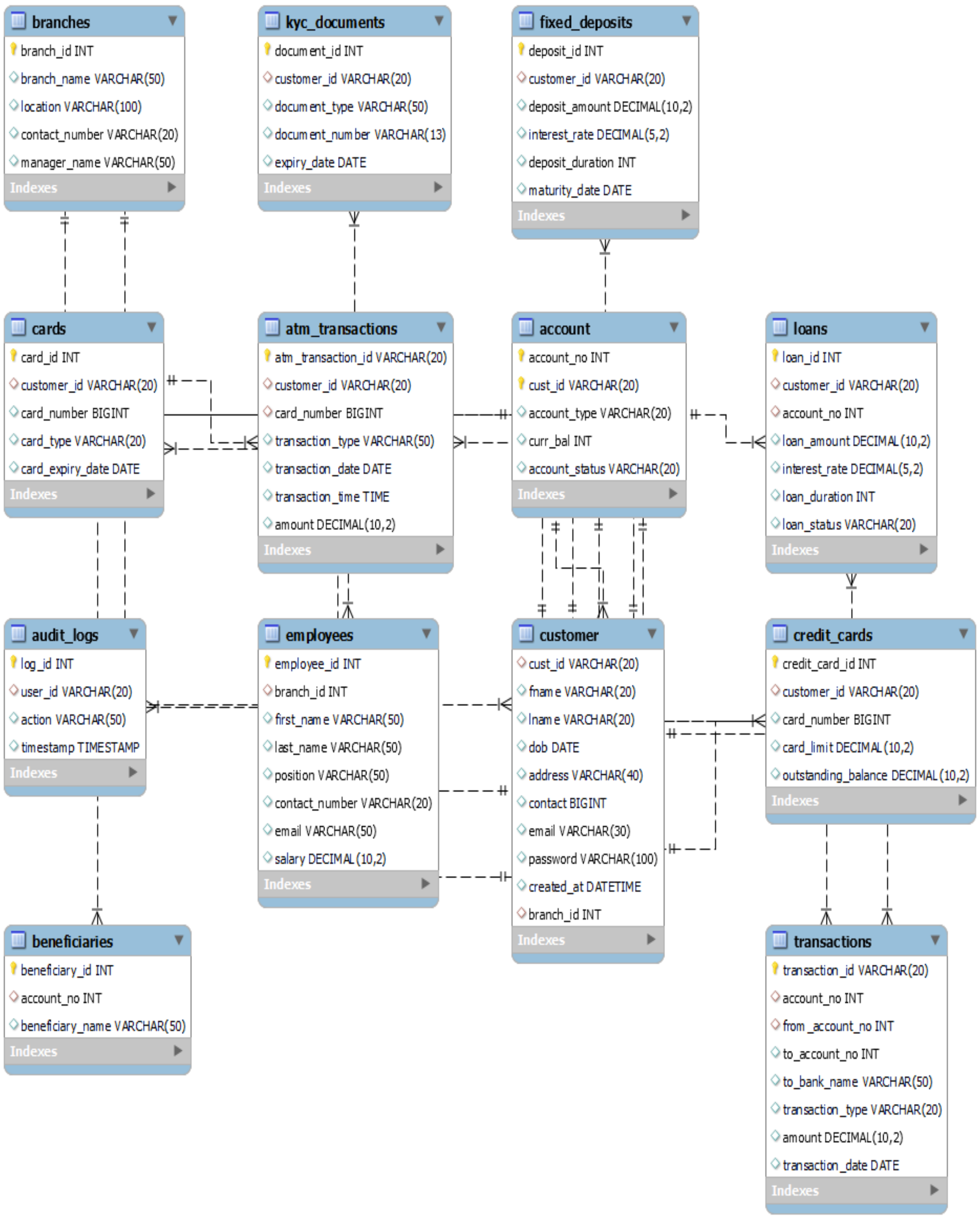
In the context of a banking system, the ER model would typically consist of the following components:

1. Entities: Entities represent the fundamental objects in the system. In a banking system, common entities include "Customer," "Account," "Transaction," "Bank," and "Branch." Each entity has its own set of attributes that describe its properties. For example, the "Customer" entity might have attributes such as "Customer ID," "First Name," "Last Name," "Date of Birth," and "Contact Details."
2. Relationships: Relationships define the associations between entities. In a banking system, common relationships include "Owns," "Performs," and "Belongs To." For instance, a customer owns one or more accounts, a transaction is performed by a customer, and an account belongs to a specific bank. Relationships can have cardinality constraints, such as one-to-one, one-to-many, or many-to-many, indicating how instances of entities are related to each other.
3. Attributes: Attributes are the properties or characteristics that describe entities. Each entity has its own set of attributes, which can be simple (e.g., name, age) or composite (e.g., address with sub-attributes like street, city, state). In the banking system, attributes for entities like "Account" could include "Account Number," "Balance," "Account Type," and "Status."
4. Primary Keys: Primary keys uniquely identify each instance of an entity. In the banking system, primary keys such as "Customer ID" for the "Customer" entity and "Account Number" for the "Account" entity ensure that each entity instance is uniquely identifiable. Primary keys are crucial for maintaining data integrity and establishing relationships between entities.

The ER model serves as a visual representation of the database design, capturing the entities, attributes, relationships, and constraints. It helps in understanding the structure of the system, identifying the data requirements, and ensuring data integrity.

Once the ER model is created, it serves as a blueprint for the actual implementation of the database using a specific database management system (DBMS) and a suitable database schema, such as the relational model. The ER model guides the creation of tables, defining the columns, data types, primary keys, foreign keys, and any constraints required to represent the entities and relationships identified in the model.

Overall, the database design and ER model for a banking system help in organizing and managing data efficiently, enabling accurate and secure storage, retrieval, and manipulation of information related to customers, accounts, transactions, and other relevant entities.


branches
branch_id INT
branch_name VARCHAR(50)
location VARCHAR(100)
contact_number VARCHAR(20)
manager_name VARCHAR(50)
Indexes
kyc_documents
document_id INT
customer_id VARCHAR(20)
document_type VARCHAR(50)
document_number VARCHAR(13)
expiry_date DATE
Indexes
fixed_deposits
deposit_id INT
customer_id VARCHAR(20)
deposit_amount DECIMAL(10,2)
interest_rate DECIMAL(5,2)
deposit_duration INT
maturity_date DATE
Indexes
cards
card_id INT
customer_id VARCHAR(20)
card_number BIGINT
card_type VARCHAR(20)
card_expiry_date DATE
Indexes
atm_transactions
atm_transaction_id VARCHAR(20)
customer_id VARCHAR(20)
card_number BIGINT
transaction_type VARCHAR(50)
transaction_date DATE
transaction_time TIME
amount DECIMAL(10,2)
Indexes
account
account_no INT
cust_id VARCHAR(20)
account_type VARCHAR(20)
curr_bal INT
account_status VARCHAR(20)
Indexes
loans
loan_id INT
customer_id VARCHAR(20)
account_no INT
loan_amount DECIMAL(10,2)
interest_rate DECIMAL(5,2)
loan_duration INT
loan_status VARCHAR(20)
Indexes
audit_logs
log_id INT
user_id VARCHAR(20)
action VARCHAR(50)
timestamp TIMESTAMP
Indexes
employees
employee_id INT
branch_id INT
first_name VARCHAR(50)
last_name VARCHAR(50)
position VARCHAR(50)
contact_number VARCHAR(20)
email VARCHAR(50)
salary DECIMAL(10,2)
Indexes
customer
cust_id VARCHAR(20)
fname VARCHAR(20)
lname VARCHAR(20)
dob DATE
address VARCHAR(40)
contact BIGINT
email VARCHAR(30)
password VARCHAR(100)
created_at DATETIME
branch_id INT
Indexes
credit_cards
credit_card_id INT
customer_id VARCHAR(20)
card_number BIGINT
card_limit DECIMAL(10,2)
outstanding_balance DECIMAL(10,2)
Indexes
beneficiaries
beneficiary_id INT
account_no INT
beneficiary_name VARCHAR(50)
Indexes
transactions
transaction_id VARCHAR(20)
account_no INT
from_account_no INT
to_account_no INT
to_bank_name VARCHAR(50)
transaction_type VARCHAR(20)
amount DECIMAL(10,2)
transaction_date DATE
Indexes

3.Normalization:

Normalization is a process in database design that organizes data in a structured and efficient manner. It eliminates data redundancy, improves data integrity, and ensures consistency in the database. The banking database project has implemented normalization techniques to achieve these goals.

The normalization process involves breaking down the database into multiple tables and establishing relationships between them. This reduces data duplication and improves data integrity by adhering to the principles of atomicity, consistency, isolation, and durability (ACID).

The banking database project has employed the following normalization techniques:

1. First Normal Form (1NF): Each table in the database represents an entity, and each attribute within the table contains only atomic values. For example, the "account" table contains attributes such as account_no, cust_id, account_type, curr_bal, and account_status, each holding a single value.
2. Second Normal Form (2NF): In addition to meeting the criteria of 1NF, all non-key attributes in each table are functionally dependent on the entire primary key. The "customer" table in the banking database project satisfies this criterion, where the attributes fname, lname, dob, address, contact, email, password, and created_at are all functionally dependent on the cust_id primary key.
3. Third Normal Form (3NF): Building upon the principles of 1NF and 2NF, all non-key attributes in each table are functionally dependent solely on the primary key and not on any other non-key attributes. This helps to eliminate transitive dependencies. The "transactions" table in the banking database project exemplifies 3NF, where attributes such as transaction_id, account_no, from_account_no, to_account_no, to_bank_name, transaction_type, amount, and transaction_date are all dependent on the primary key.

By following the normalization process, the banking database project ensures data integrity, minimizes data redundancy, and allows for efficient data retrieval and manipulation. The normalized structure also facilitates the establishment of relationships between tables through primary key-foreign key associations, enabling the implementation of various functionalities such as data retrieval, updates, and reporting.

4. Database Functionality:

Database functionality refers to the set of operations and features that a database management system (DBMS) provides to effectively store, retrieve, manipulate, and manage data. These functionalities ensure the reliability, security, and efficiency of the database system. Some common database functionalities include:

1. Data Storage: The DBMS stores data in an organized and structured manner, using tables, columns, and rows. It manages the physical storage of data on disk or in memory.
2. Data Retrieval: Users can query the database to retrieve specific data based on certain criteria using SQL (Structured Query Language) or other query languages. The DBMS performs efficient data retrieval using indexing, caching, and optimization techniques.
3. Data Manipulation: The DBMS allows users to insert, update, and delete data in the database. These operations maintain data integrity and consistency.
4. Data Security: The DBMS provides mechanisms to ensure data security and access control. It includes authentication, authorization, and encryption techniques to protect sensitive information from unauthorized access or modification.
5. Data Integrity: The DBMS enforces data integrity constraints, such as primary keys, foreign keys, unique constraints, and check constraints, to maintain the accuracy and consistency of data.
6. Transactions: The DBMS supports transaction management, ensuring the atomicity, consistency, isolation, and durability (ACID properties) of database operations. It allows multiple operations to be treated as a single unit, ensuring data consistency even in the presence of failures.
7. Concurrency Control: The DBMS manages concurrent access to the database by multiple users or applications, preventing conflicts and ensuring data consistency. It uses locking, timestamping, or other techniques to control access and maintain data integrity.
8. Indexing and Query Optimization: The DBMS creates indexes on frequently accessed columns to improve query performance. It also optimizes query execution plans to minimize the time and resources required for data retrieval.
9. Backup and Recovery: The DBMS provides mechanisms to backup the database periodically and recover it in case of system failures or data loss. It ensures data availability and reliability.

5. Query Examples

1. Retrieve all customer details (first name, last name, email) who have a savings account with an account balance greater than 5000.

2. Get the total number of active accounts in the database.

3. List the customers who made a transaction of more than 1000 on a specific date.

4. Find the account numbers with the highest and lowest current balances.

5. Get the customer details who have a fixed deposit with a maturity date within the next 30 days.

6. Retrieve the details of customers who have a credit card with an outstanding balance greater than 500.

7. Get the total number of transactions of each transaction type.

8. List the customers who have accounts in multiple banks.

9. Find the average account balance for each account type.

10. Retrieve the customer details along with their branch information.

11. Retrieve the total balance for each customer along with their account type, sorted in descending order of the total balance.

12. Get the customer details who have the highest account balance for each account type.

13. Retrieve the customers who have made transactions with a specific bank, along with the total amount of transactions made by each customer.

14. Get the customer details who have not made any transactions.

15. Retrieve the customer details along with their account type, showing "Active" if the account status is 'Active' and "Inactive" otherwise.

16. Retrieve the customers who have at least two active accounts of different types.

17. Get the customers who have made transactions exceeding the average transaction amount, along with the total transaction count for each customer.

18. Retrieve the customers who have a loan amount greater than the average loan amount of all customers, along with the maximum loan amount.

19. Get the customers who have the same last name and display the total balance across all their accounts.

20. Retrieve the customers who have made a cash withdrawal transaction within the last 30 days, along with the count of such transactions.

21. Get the count of transactions for each day in a specific month: