Banking System DataBase

- **Project Title:** Banking System DataBase
- **Student Name**: Meron Goitom
- **Student Id**: 9225226654
- **Github username**: merongoitom

Checkpoint #	Date Submitted
Checkpoint#3	04/01/2024

Section II: Table Contents

Section III: Project Description

- Banking Database Management System

The motivation behind creating this database system is to streamline banking operations, enhance customer service, and improve data management efficiency. Traditional banking processes often involve manual paperwork, redundant data entry, and limited accessibility to customer information. By implementing a centralized database system, we aim to address these challenges by digitizing processes, enabling real-time access to customer data, and automating routine tasks. This database system will also help comply with regulatory requirements, analyze customer behavior, and identify new revenue generation opportunities.

Our database system is designed to serve as a comprehensive platform for managing all aspects of banking operations. It consists of multiple interconnected modules that handle customer management, account operations, transaction processing, loan management, and reporting. The system utilizes a relational database model to organize and store data efficiently, ensuring data integrity and consistency. With user-friendly interfaces and intuitive workflows, bank staff can easily navigate the system to perform tasks such as account opening, transaction processing, and customer support. The system also incorporates robust security measures to protect sensitive customer information and prevent unauthorized access.

- Unique Features

1. Advanced Analytics: It also manages the system and integrates powerful analytics tools that leverage customer data to generate insights and predictive models. By analyzing transaction patterns, customer behavior, and market trends, banks can identify opportunities for personalized offerings, risk mitigation, and revenue optimization.

2. Omni-Channel Integration: Unlike traditional banking systems, our database system offers seamless integration across multiple channels, including online banking, mobile banking, ATMs, and branch offices. Customers can initiate transactions through any channel and experience consistent service quality, enhancing their overall banking experience.

3. Automated Compliance: Our database system incorporates automated compliance checks and reporting features to ensure adherence to regulatory requirements such as KYC (Know Your Customer), AML (Anti-Money Laundering), and GDPR (General Data Protection Regulation). By automating compliance processes, banks can reduce manual errors, minimize regulatory risks, and streamline audit procedures.

- Software Tools Benefiting from Our Database System:

1. Salesforce Financial Services Cloud: Salesforce Financial Services Cloud is a popular CRM platform banks use to manage customer relationships and sales activities. By integrating with our database system, Financial Services Cloud can access real-time customer data, transaction history, and account information, enabling personalized interactions, targeted marketing campaigns, and improved customer engagement.

2. Oracle Financial Services Analytical Applications (OFSAA): OFSAA is a comprehensive suite of analytical applications designed for financial institutions to perform risk management, profitability analysis, and regulatory reporting. By leveraging data from our database system, OFSAA can enhance its analytical capabilities, enabling banks to gain deeper insights into their operations, optimize capital allocation, and meet regulatory compliance requirements more effectively.

Section IV

● Functional Database Requirements

1. Customer

1.1. A customer shall have at most one address.

1.2. A customer shall have a multiple loan

1.3. A customer shall be able to transfer multiple funds between their accounts.

1.4. A customer can have one or more accounts

1.5. Each customer shall be able to view only their savings account

1.6. Each customer shall be able to view only their checking account.

1.7. Each customer can create multiple saving accounts

1.8. Customers shall be able to get service from multiple Branch

1.9. A customer can request multiple credit cards.

1.10 A customer can request multiple debit cards.

1,11 Customers can receive services from multiple employees

2. Account

2.1. Account shall be able to be created by multiple customers

2.2. An account can have multiple interest rates.

2.3. An account shall have multiple loans.

2.4. An account shall have one or more customers

2.5. Account shall be either a savings account, checking account, or Joint account

2.6. Account has only on financial credit score.

2.7 . An account can have multiple account statuses.

3. Branch

3.1. A branch can be associated with multiple employees.

3.2. A branch shall serve multiple customers at a time.

3.3 A branch shall have multiple ATMs.

3.4. A branch should only be associated with the respective bank.

3.5. A branch can managed by one or more managers.

4. Loan

4.1. A loan shall have at least a Customer ID

4.2. A loan can be associated with multiple account holders.

4.3 . A loan shall have multiple transactions.

5. Employee

5.1. Each employee can give service to multiple customers.

5.2. Employees can work with related jobs at multiple branches.

5.3. Employees can only be employed at one bank

6.Credit Card

6.1. A credit card must be capable of recording multiple transactions.

6.2. A credit card can be created by many customers.

7. Transaction

7.1. A transaction shall be able to be associated with only one specific account whether checking or saving

7.2. A transaction can be associated with only one loan

7.3. A Transaction is associated with only one credit card.

7.4. Transaction can only be initiated at one ATM at a time

8. ATM

8.1. An ATM shall be linked to one branch.

8.2. ATM can initiate multiple transactions.

9. Bank

9.1. A bank should be able to operate multiple branches.

9.2. A bank can employ many employees.

10. Saving Account

10.1. Savings account balance allowing to view only to the associated customer.

10.2 A savings account shall be able created by many customers.

11. Checking Account

11.1. Checking account balances allows one to view only the associated customer.

11.2. A checking account shall have multiple transactions

11.3. A checking account shall be able created by many customers.

12. Interest Rate

12.1. interest rate can be associated with multiple accounts.

13. Address

13.1. The address shall have at most one customer ID

14. Credit Score

14.1. Credit score should associated with only one account

15. Account Status

15.1. Account status can be associated with multiple accounts.

16. Check

16.1. Check should be able associated only one payment

17. Payment

17.1. Payment should be able to have only one check.

17.2. Payment can be associated with multiple mortgages

17.3. Payment can be made using multiple debit cards.

18. Manager

18.1 A Manager can manage only one branch

19. Debit Card

19.1. A debit card can be created by multiple customers.

19.2. Each debit card can be used for multiple payments.

20. Mortgage

20.1 Each mortgage can have multiple payments.

Section V

- # Non-Functional Database Requirements

1. Performance:

1.1. Response Time: The system shall respond to user queries and transactions within 2 seconds on average.

1.2. Transaction Throughput: The database system shall support a minimum of 1000 transactions per second during peak hours.

1.3. Batch Processing: Batch processing jobs shall be completed within 1 hour for daily reconciliation and reporting tasks.

1.4. Indexing: Indexes shall be optimized to minimize query execution time and improve overall system performance.

1.5. Data Retrieval: Queries involving complex joins and aggregations shall return results within 5 seconds to ensure timely access to information.

2. Security

2.1. Access Control: Role-based access control (RBAC) shall be implemented to restrict access to sensitive data based on user roles and privileges.

2.2. Encryption: Data encryption shall be enforced for all sensitive information stored in the database, including customer credentials and financial transactions.

2.3. Audit Trail: The system shall maintain a comprehensive audit trail of database activities, including user logins, modifications, and access attempts.

2.4. Intrusion Detection: Intrusion detection and prevention systems (IDPS) shall be deployed to monitor database traffic and detect suspicious activities or unauthorized access attempts.

2.5. Data Masking: Sensitive data such as account numbers and social security numbers shall be masked in database queries and reports to protect confidentiality.

3. Scalability:

3.1. Horizontal Scaling: The database architecture shall support horizontal scaling by adding more database nodes to distribute the workload and accommodate growth.

3.2. Load Balancing: Load balancing mechanisms shall be implemented to evenly distribute database requests across multiple servers for optimal performance.

3.3. Sharding: Data sharding techniques shall be employed to partition large datasets across multiple servers to improve scalability and performance.

3.4. Auto-scaling: The database system shall support auto-scaling capabilities to dynamically adjust resource allocation based on demand to handle fluctuating workloads.

3.5. Partitioning: Data partitioning strategies shall be implemented to divide data logically and distribute it across storage devices for efficient management and scalability.

4. Capability:

4.1. Data Integrity: The database system shall enforce data integrity constraints to ensure the accuracy and consistency of stored information.

4.2. Backup and Recovery: Regular backups shall be performed to safeguard data integrity, and robust recovery procedures shall be in place to restore data in case of failures.

4.3. Replication: Data replication shall be implemented to create redundant copies of critical data for high availability and disaster recovery purposes.

4.4. Concurrency Control: Concurrency control mechanisms shall be employed to manage concurrent access to data and prevent data corruption or inconsistencies.

4.5. Compliance: The database system shall comply with industry regulations such as GDPR, PCI DSS, and SOX to protect customer data and ensure regulatory compliance.

5. Environmental:

5.1. Energy Efficiency: The data center hosting the database servers shall be designed for energy efficiency, utilizing energy-efficient hardware and cooling systems.

5.2. Cooling Systems: The data center shall be equipped with efficient cooling systems to maintain optimal operating temperatures and minimize environmental impact.

5.3. Green Computing: The database system shall prioritize energy-efficient computing practices and use renewable energy sources where possible to reduce carbon footprint.

5.4. E-waste Management: Proper e-waste management practices shall be followed for disposing of obsolete hardware and electronic components responsibly.

5.5. Compliance with Environmental Regulations: The database system shall comply with environmental regulations and standards regarding electronic waste disposal and energy conservation.

6. Coding Standards:

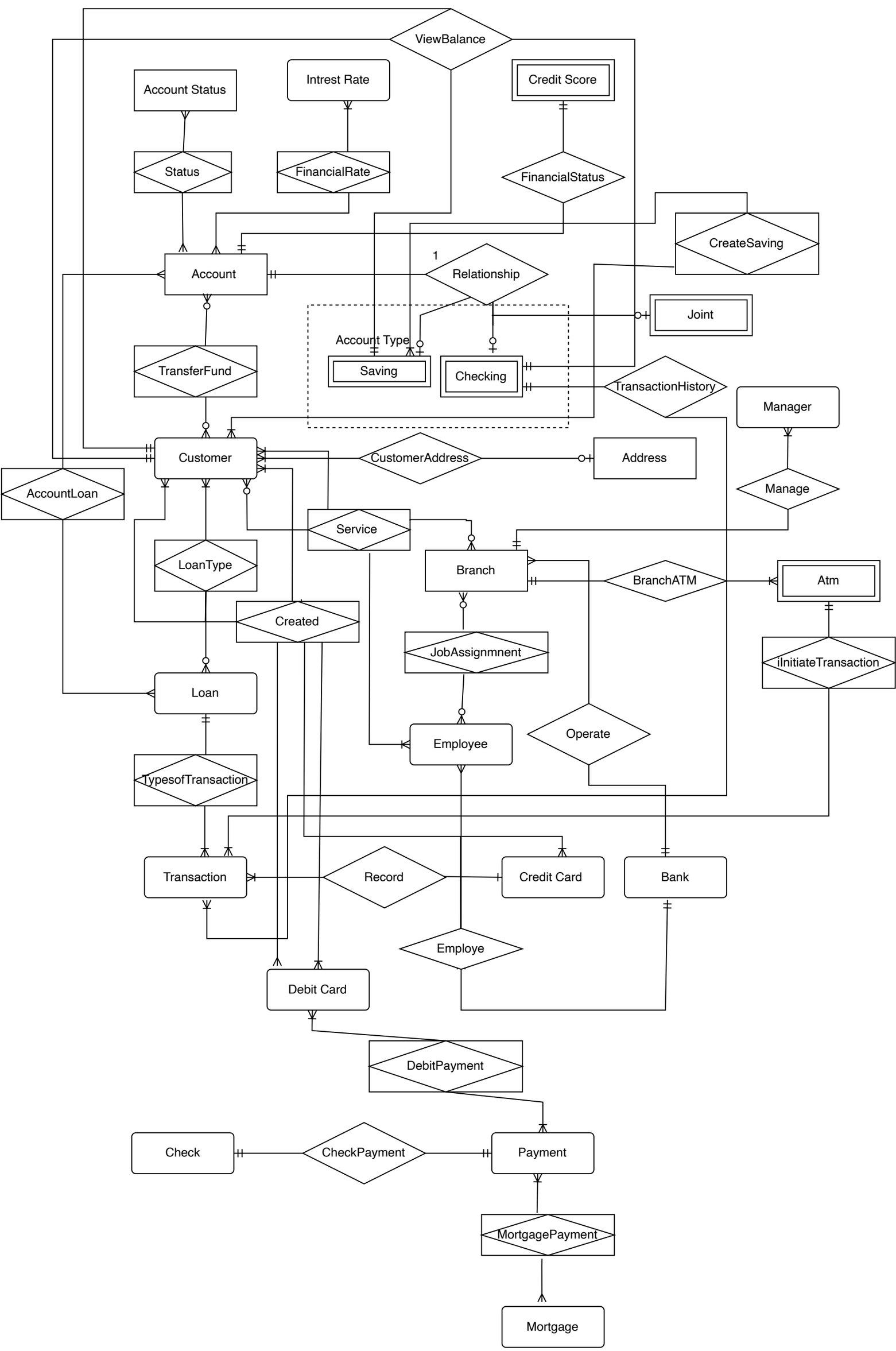
6.1. Naming Conventions: Database objects such as tables, columns, and stored procedures shall follow consistent naming conventions for clarity and maintainability.

6.2. Documentation: The database system shall be well-documented, including data dictionaries, schema diagrams, and code comments, to facilitate understanding and future development.

6.3. Error Handling: Robust error handling mechanisms shall be implemented to handle exceptions gracefully and prevent data loss or corruption.

6.4. Code Review: Database code shall undergo regular peer reviews to ensure adherence to coding standards, identify potential issues, and maintain code quality.

6.5. Version Control: Changes to database schema and code shall be managed using version control systems to track revisions and facilitate collaboration among developers.


ViewBalance
Account Status
Intrest Rate
Credit Score
Status
FinancialRate
FinancialStatus
CreateSaving
Account
1
Relationship
Joint
Account Type
Saving
Checking
TransferFund
TransactionHistory
Manager
Customer
CustomerAddress
Address
AccountLoan
Manage
Service
LoanType
Branch
BranchATM
Atm
Created
JobAssignmnent
iInitiateTransaction
Loan
Operate
Employee
TypesofTransaction
Transaction
Record
Credit Card
Bank
Employe
Debit Card
DebitPayment
Check
CheckPayment
Payment
MortgagePayment
Mortgage

Section VII

1. Customer (Strong)
   customer_id: key, numeric
   name: composite, alphanumeric
   date_of_birth: multivalue, timestamp

2. Account (Strong)
   account_number: key, alphanumeric
   account_type: composite, alphanumeric
   balance: multivalue, numeric

3. Branch (Strong)
   * branch_id: key, numeric
   * branch_name: composite, alphanumeric
   * address: multivalue, alphanumeric

4. Loan (Strong)
   *loan_id: key, alphanumeric
   * loan_type: composite, alphanumeric
   * amount: multivalue, numeric

5. Employee (Strong)
   * employee_id: key, numeric
   * name: composite, alphanumeric
   * hire_date: multivalue, timestamp

6. Credit Card (Strong)
   * card_number: key, alphanumeric
   * card_type: composite, alphanumeric
   * expiry_date: multivalue, timestamp

7. Transaction (Strong)
   * transaction_id: key, alphanumeric
   * transaction_type: composite, alphanumeric
   * amount: multivalue, numeric

8. ATM (Weak)
   * atm_id: key, alphanumeric

  * Location: composite, alphanumeric
  * installation_date: multivalue, timestamp

9. Bank (Strong)
  * bank_id: key, numeric
  * bank_name: composite, alphanumeric
  * bank_address: multivalue, alphanumeric

10. Saving Account (weak)
  * saving_number: key, alphanumeric
  * account_holder_name: composite, alphanumeric
  * opening_date: multivalue, timestamp

11. Checking Account (weak)
  * checking_number: key, alphanumeric
  * account_holder_name: composite, alphanumeric
  * opening_date: multivalue, timestamp

12. Interest Rate (Strong)
  * rate_id: key, alphanumeric
  * rate_value: composite, numeric
  * effective_date: multivalue, timestamp

13. Address (Strong)
  * address_id: key, numeric
  * street: composite, alphanumeric
  * city: composite, alphanumeric
  * postal_code: composite, alphanumeric

14. Credit Score (weak)
  * score_id: key, alphanumeric
  * score_value: composite, numeric
  * assessment_date: multivalue, timestamp

15. Account Status (Strong)
  * status_id: key, alphanumeric

  * status_description: composite, alphanumeric
  * last_updated_date: multivalue, timestamp

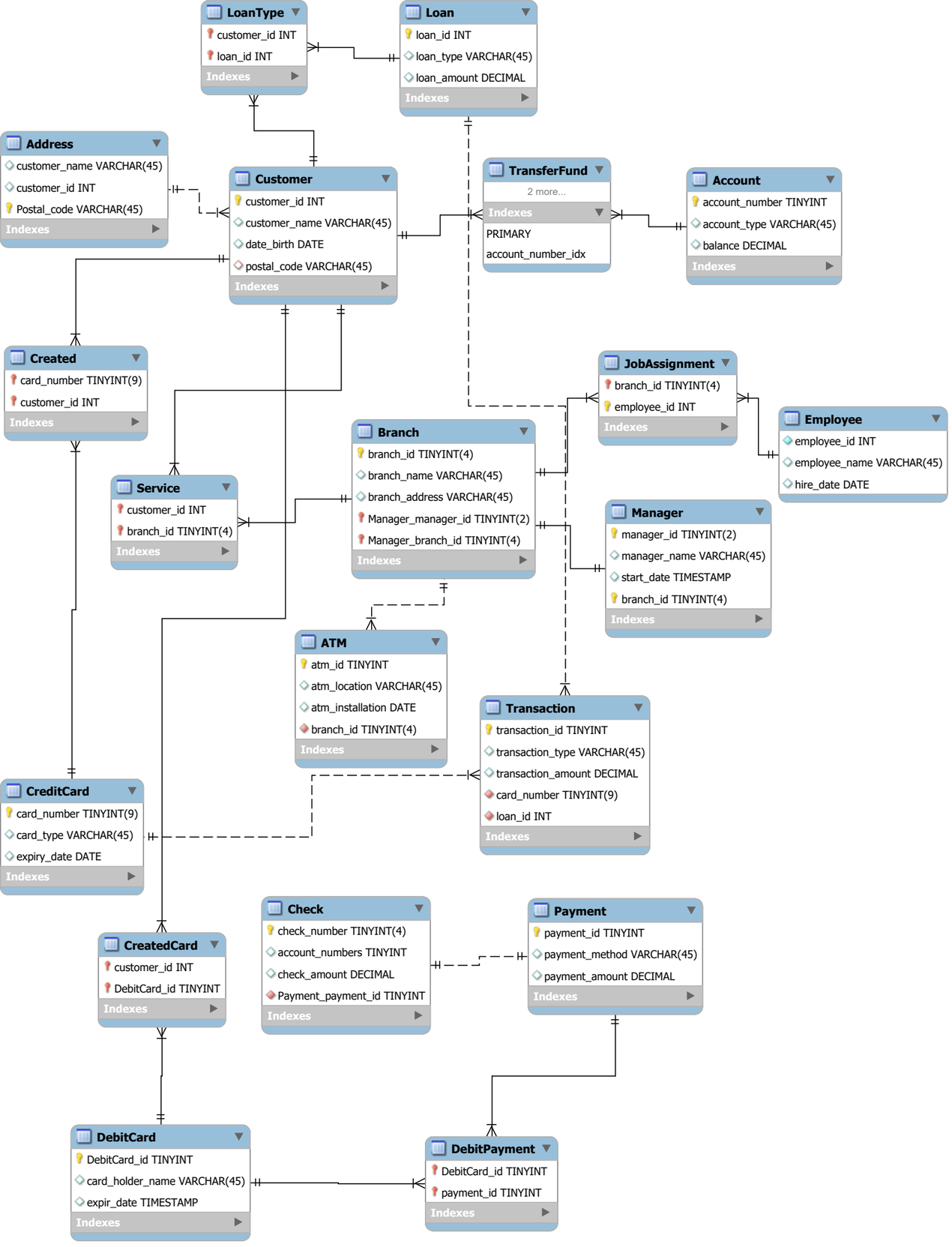
16. Check (Strong)
  * check_number: key, alphanumeric
  * account_number: composite, alphanumeric
  * amount: multivalue, numeric

17. Payment (Strong)
  * payment_id: key, alphanumeric
  * payment_method: composite, alphanumeric
  * amount: multivalue, numeric

18. Manager (Strong)
  * manager_id: key, alphanumeric
  * manager_name: composite, alphanumeric
  * start_date: multivalue, timestamp

19. Debit Card (Strong)
  * Debit_card_number: key, alphanumeric
  * card_holder_name: composite, alphanumeric
  * expiration_date: multivalue, timestamp

20. Mortgage (Strong)
  * mortgage_id: key, alphanumeric
  * loan_amount: composite, numeric
  * start_date: multivalue, timestamp


LoanType
customer_id INT
loan_id INT
Indexes
Loan
loan_id INT
loan_type VARCHAR(45)
loan_amount DECIMAL
Indexes
Address
customer_name VARCHAR(45)
customer_id INT
Postal_code VARCHAR(45)
Indexes
Customer
customer_id INT
customer_name VARCHAR(45)
date_birth DATE
postal_code VARCHAR(45)
Indexes
TransferFund
2 more...
Indexes
PRIMARY
account_number_idx
Account
account_number TINYINT
account_type VARCHAR(45)
balance DECIMAL
Indexes
Created
card_number TINYINT(9)
customer_id INT
Indexes
JobAssignment
branch_id TINYINT(4)
employee_id INT
Indexes
Employee
employee_id INT
employee_name VARCHAR(45)
hire_date DATE
Service
customer_id INT
branch_id TINYINT(4)
Indexes
Branch
branch_id TINYINT(4)
branch_name VARCHAR(45)
branch_address VARCHAR(45)
Manager_manager_id TINYINT(2)
Manager_branch_id TINYINT(4)
Indexes
Manager
manager_id TINYINT(2)
manager_name VARCHAR(45)
start_date TIMESTAMP
branch_id TINYINT(4)
Indexes
ATM
atm_id TINYINT
atm_location VARCHAR(45)
atm_installation DATE
branch_id TINYINT(4)
Indexes
Transaction
transaction_id TINYINT
transaction_type VARCHAR(45)
transaction_amount DECIMAL
card_number TINYINT(9)
loan_id INT
Indexes
CreditCard
card_number TINYINT(9)
card_type VARCHAR(45)
expiry_date DATE
Indexes
Check
check_number TINYINT(4)
account_numbers TINYINT
check_amount DECIMAL
Payment_payment_id TINYINT
Indexes
Payment
payment_id TINYINT
payment_method VARCHAR(45)
payment_amount DECIMAL
Indexes
CreatedCard
customer_id INT
DebitCard_id TINYINT
Indexes
DebitCard
DebitCard_id TINYINT
card_holder_name VARCHAR(45)
expir_date TIMESTAMP
Indexes
DebitPayment
DebitCard_id TINYINT
payment_id TINYINT
Indexes

Section IX

Table	FK	On Delete	On Update	Comment
Address	Customer	ON CASCADE	ON CASCADE	The Address referring to the customer must be deleted.
Loan	LoanType	NO ACTION	NO ACTION	Prevent deletion and data loss because it has a many-to-many relation
Customer	LoanType	NO ACTION	NO ACTION	Prevent deletion and data loss because it has a many-to-many relation
Customer	TransferFund	NO ACTION	NO ACTION	Prevent deletion and data loss because it has a many-to-many relation
Account	TransferFund	RESTRICT	RESTRICT	Prevent deletion and data loss because it has a many-to-many relation
Customer	Created	NO ACTION	NO ACTION	Prevent deletion and data loss because it has a many-to-many relation
CreditCard	Created	NO ACTION	NO ACTION	Prevent deletion and data loss because it has a many-to-many relation
CreditCard	Transaction	NO ACTION	NO ACTION	One-to-many relation to avoid data loss and deletion
Customer	Branch	NO ACTION	NO ACTION	Prevent deletion and data loss because it has a many-to-many relation
Branch	ATM	RESTRICT	RESTRICT	One-to-many relation to avoid data loss and deletion. If the ATM is deleted we can not keep the branch data
Branch	JobAssignment	RESTRICT	RESTRICT	Keep the date on the Branch table because it has many- to many- relation
Employee	JobAssignment	NO ACTION	NO ACTION	It keeps the employee data the same even if the employee's job assignment changes
Check	Payment	ON CASCADE	ON CASCADE	The payment refers to the checking must be deleted
Branch	Manager	ON CASCADE	ON CASCADE	If manager deleted from branch must be deleted from Manager because one to one relation