Databases - Progetto 'Database Banca'

Davide Zicca

May 14, 2021

L'obiettivo di tale progetto è quello di simulare alcune operazioni tipiche che vengono espletate da una banca commerciale che opera attraverso filiali (*Branches*) dislocate su vari territori. Vengono, dunque, tralasciate le operazioni di investimento in fondi a gestione passiva (o attiva) ed altre operazioni che tipicamente una banca commerciale svolge.

1 Contenuto e logica di funzionamento del Database

- Tale Banca opera sul territorio utilizzando N filiali. Ogni filiale ha un identificatore univoco, un nome e un indirizzo. Una filiale opera per un numero arbitrario di clienti.
- Ogni conto bancario ha un identificatore univoco e il suo saldo corrente. Ad ogni account è associata una carta attiva e la carta contiene il suo numero univoco, la data di scadenza e se la carta è stata bloccata o meno. Un account può anche avviare prestiti e altre transazioni.
- Il database tiene anche traccia dei prestiti; e ogni prestito ha un identificatore univoco, il tipo di prestito erogato che può essere ad esempio: prestiti personali, prestiti per acquisto auto, etc., l'importo di denaro che il cliente ha già rimborsato e in aggiunta la data di inizio e la somma dovuta del prestito. Oltre a questo, ogni tipo di prestito ha un identificativo, un nome, una breve descrizione, un importo di base e un tasso di interesse di base. Inoltre, il database contiene *record* di transazioni, ciascuna con un unico file identificativo, una descrizione della transazione e l'importo e la data del trasferimento.
- Ogni filiale ha dei clienti con attributi caratteristici. Ogni cliente ha un identificativo univoco, un nome e un cognome, la data di nascita e il suo genere (Uomo/Donna - *Male/Female*). Un cliente amministra anche uno o più conti bancari.
- Creazione Users e aggiunta dati per ogni cliente.
- Verifica che ogni account abbia un saldo minimo di disponibilità di denaro in ogni momento

2 Queries introdotte

1. Query 1: Numero di clienti che hanno un account in due o più filiali.
2. Query 2: Alla fine di ogni anno, un documento attestante tutti i movimenti è generato per ogni account.
3. Query 3: Cerca clienti che non hanno account aperti.

3 MySQL: codice implementato

Di seguito, il codice completo che è stato implementato in MySQL WorkBench:

```
# Davide Zicca - Bank DB Project
-- -----------------------------------------------
-- Creo Schema e Tabelle
-- -----------------------------------------------

CREATE SCHEMA Bank;

-- -----------------------------------------------
USE Bank;
CREATE TABLE Branch (
id INT,
name CHAR(50) UNIQUE,
address CHAR(50),
PRIMARY KEY (id)
);

-- -----------------------------------------------
USE Bank;
CREATE TABLE Card (
id INT,
number CHAR(50) UNIQUE,
expiration_date DATE,
is_blocked BOOL,
PRIMARY KEY (id)
);


-- -----------------------------------------------
USE Bank;
CREATE TABLE Loan_type (
```

```
id INT,
type CHAR(50) UNIQUE,
description CHAR(100),
base_amount DECIMAL(10, 3),
base_interest_rate DECIMAL(10, 3),
PRIMARY KEY (id)
);

-- ------------------------------------------------
USE Bank;
CREATE TABLE Customer (
id INT,
branch_id INT,
first_name CHAR(50),
last_name CHAR(50),
date_of_birth DATE,
gender CHAR(6),
PRIMARY KEY (id),
FOREIGN KEY (branch_id) REFERENCES Branch(id)
ON UPDATE CASCADE
ON DELETE SET NULL
);

-- ------------------------------------------------
USE Bank;
CREATE TABLE Account (
id INT,
customer_id INT,
card_id INT,
balance CHAR(50),
PRIMARY KEY (id),
FOREIGN KEY (customer_id) REFERENCES Customer(id)
ON UPDATE CASCADE
ON DELETE SET NULL,
FOREIGN KEY (card_id) REFERENCES Card(id)
ON UPDATE CASCADE
ON DELETE SET NULL
);

-- ------------------------------------------------
USE Bank;
CREATE TABLE Loan (
id INT,
account_id INT,
```

```
loan_type_id INT,
amount_paid DECIMAL(10, 3),
start_date DATE,
due_date DATE,
PRIMARY KEY (id),
FOREIGN KEY (account_id) REFERENCES Account(id)
ON UPDATE CASCADE
ON DELETE SET NULL,
FOREIGN KEY (loan_type_id) REFERENCES Loan_type(id)
ON UPDATE CASCADE
ON DELETE SET NULL
);

-- ------------------------------------------------
USE Bank;
CREATE TABLE Transaction (
id INT,
account_id INT,
description CHAR(100),
amount DECIMAL(10, 3),
date DATE,
PRIMARY KEY (id),
FOREIGN KEY (account_id) REFERENCES Account(id)
ON UPDATE CASCADE
ON DELETE SET NULL
);

-- ------------------------------------------------
-- Creo gli Users
-- ------------------------------------------------
CREATE USER 'mariod'@'%' IDENTIFIED BY 'password';
CREATE USER 'gallagherL'@'%' IDENTIFIED BY 'password';
CREATE USER 'batiG'@'%' IDENTIFIED BY 'password';
GRANT ALL ON *.* TO 'mariod'@'%';
GRANT ALL ON *.* TO 'gallagherL'@'%' WITH GRANT OPTION;
GRANT SELECT, UPDATE, DELETE ON *.* TO 'batiG'@'%';
SELECT * FROM mysql.user;

SHOW GRANTS for 'batiG'@'%';

-- ------------------------------------------------
-- Creo una visualizzazione
-- ------------------------------------------------
USE Bank;
```

```
CREATE VIEW User_role_information AS
SELECT User, Select_priv, Insert_priv, Update_priv, Delete_priv, Create_priv
FROM mysql.user
WHERE Select_priv = 'Y' OR Insert_priv = 'Y' OR Update_priv = 'Y' OR Delete_priv = 'Y' OR

-- ------------------------------------------------
-- Inserisco valori nel Database
-- ------------------------------------------------
USE Bank;
INSERT INTO Branch (id, name, address) VALUES ('1', 'Roggiano Bank', 'Roggiano');
INSERT INTO Branch (id, name, address) VALUES ('2', 'San Marco Argentano Bank', 'San Marc
INSERT INTO Branch (id, name, address) VALUES ('3', 'Cosenza Bank', 'Cosenza');
INSERT INTO Branch (id, name, address) VALUES ('4', 'Foggia Bank', 'Foggia');
INSERT INTO Branch (id, name, address) VALUES ('5', 'Firenze Bank', 'Firenze');

-- ------------------------------------------------
USE Bank;
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES ('1', '1234567890123456
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES ('2', '1234567890123457
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES ('3', '1234567890123458
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES ('4', '1234567890123459
INSERT INTO Card (id, number, expiration_date, is_blocked) VALUES ('5', '1234567890123450

-- ------------------------------------------------
USE Bank;
INSERT INTO Loan_type (id, type, description, base_amount, base_interest_rate) VALUES ('1
INSERT INTO Loan_type (id, type, description, base_amount, base_interest_rate) VALUES ('2
INSERT INTO Loan_type (id, type, description, base_amount, base_interest_rate) VALUES ('3
INSERT INTO Loan_type (id, type, description, base_amount, base_interest_rate) VALUES ('4
INSERT INTO Loan_type (id, type, description, base_amount, base_interest_rate) VALUES ('5

-- ------------------------------------------------
USE Bank;
INSERT INTO Customer (id, branch_id, first_name, last_name, date_of_birth, gender) VALUES
INSERT INTO Customer (id, branch_id, first_name, last_name, date_of_birth, gender) VALUES
INSERT INTO Customer (id, branch_id, first_name, last_name, date_of_birth, gender) VALUES
INSERT INTO Customer (id, branch_id, first_name, last_name, date_of_birth, gender) VALUES
INSERT INTO Customer (id, branch_id, first_name, last_name, date_of_birth, gender) VALUES

-- ------------------------------------------------
USE Bank;
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('1', '1', '1', '100000')
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('2', '2', '2', '10000');
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('3', '3', '3', '200');
```

```
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('4', '5', '4', '500000')
INSERT INTO Account (id, customer_id, card_id, balance) VALUES ('5', '5', '5', '1000000')

-- ------------------------------------------------
USE Bank;
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid, start_date, due_date) VALUES
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid, start_date, due_date) VALUES
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid, start_date, due_date) VALUES
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid, start_date, due_date) VALUES
INSERT INTO Loan (id, account_id, loan_type_id, amount_paid, start_date, due_date) VALUES

-- ------------------------------------------------
USE Bank;
INSERT INTO Transaction (id, account_id, description, amount, date) VALUES ('1', '1', 'De
INSERT INTO Transaction (id, account_id, description, amount, date) VALUES ('2', '2', 'De
INSERT INTO Transaction (id, account_id, description, amount, date) VALUES ('3', '5', 'De
INSERT INTO Transaction (id, account_id, description, amount, date) VALUES ('4', '5', 'De
INSERT INTO Transaction (id, account_id, description, amount, date) VALUES ('5', '5', 'De

-- ------------------------------------------------
-- Verifica che ogni account abbia un saldo minimo di disponibilità di denaro in ogni mon
-- ------------------------------------------------
USE Bank;
delimiter //
CREATE TRIGGER bal_limit_insert BEFORE INSERT ON Account FOR EACH ROW
BEGIN
DECLARE message varchar(50);
IF NEW.balance < 100 THEN
SET message= CONCAT('Errore di inserimento: nuovo saldo troppo basso: ', NEW.balance);
SIGNAL SQLSTATE '46000'
SET MESSAGE_TEXT = message;
END IF;
END;
//

CREATE TRIGGER bal_limit_update BEFORE UPDATE ON Account FOR EACH ROW
BEGIN
DECLARE message varchar(50);
IF NEW.balance < 100 THEN
SET message= CONCAT('Errore di update: nuovo saldo troppo basso: ', NEW.balance);
SIGNAL SQLSTATE '46000'
SET MESSAGE_TEXT = message;
END IF;
END;
```

```
//
delimiter ;

-- -----------------------------------------------
--
-- -----------------------------------------------
-- 1. Numero di clienti che hanno un account in due o più filiali.
-- -----------------------------------------------
USE Bank;

SELECT c.first_name, c.last_name
FROM Customer c
WHERE c.id IN (SELECT customer_id
FROM Customer_Branch cb
GROUP BY customer_id
HAVING COUNT(*) >= 2);


-- -----------------------------------------------
-- 2.  Alla fine di ogni anno, un documento attestante tutti i movimenti è generato per c
-- -----------------------------------------------
CREATE EVENT IF NOT EXISTS Account_transactions_every_year
ON SCHEDULE AT '2021-12-31' + INTERVAL 1 year
DO SELECT *
FROM Transaction t

-- -----------------------------------------------
-- 3. Cerca clienti che non hanno account aperti.
-- -----------------------------------------------
USE Bank;
SELECT c.first_name, c.last_name
FROM Customer c
WHERE c.id NOT IN (SELECT customer_id
FROM Account cb
GROUP BY customer_id);

-- -----------------------------------------------
--     --- Fine ---
-- -----------------------------------------------
DROP DATABASE Bank;
```