Project Milestone #2

Logical Modelling using Relational Model

Introduction

This section outlines the logical design of the Investment Bank Back-Office Database, presented as a formal relational model. This model serves as the logical blueprint for implementation and is a direct, systematic translation of the conceptual Enhanced Entity-Relationship (EER) diagram created in the previous milestone.

Process Description

The primary objective of this phase was to convert the EER diagram's conceptual constructs into the concrete structures of the relational model (tables, columns, and keys). This was achieved through a set of mapping rules:

- **Entities to Relations**: Each strong entity from the EER diagram (e.g., Client, Account, Trade) has been mapped directly to a relation (table).
- **Attributes to Columns**: The attributes of each entity (e.g., first_name, price) have become the columns of their respective relations.
- **Relationships to Foreign Keys**: The relationships between entities are now enforced through foreign keys. For example, the one-to-many "owns" relationship between Client and Account is implemented by placing client_id (the primary key of Client) as a foreign key in the Account relation.
- **Keys**: Primary keys are identified with an underline, and foreign keys are shown in *italics* to clearly denote the links and integrity constraints between relations.

Specific EER constructs were also systematically mapped. The specialization hierarchy for Security was translated by creating separate relations (Bond, Equity) for the subclasses, each containing a foreign key that references the Security superclass. Composite primary keys were established for relations like Position and MarketData to ensure entity integrity based on their identifying attributes, as defined in the EERD.

The resulting schema, detailed below, formally defines the database structure and its integrity constraints, providing the final blueprint before proceeding to physical implementation with SQL.

Relational Model

Client (client_id, first_name, middle_name, last_name, country, client_type, client_status, tax_id)

User (user_id, department, role, email, last_login)

Account (account_id, *client_id*, opening_date, account_type, status, *manager_id*)

- *client_id* refers to client_id in Client, NOT NULL.
- *manager_id* refers to user_id in User, NOT NULL.

Counterparty (counterparty_id, counterparty_name, counterparty_type, credit_limit, settlement_instructions)

Security (cusip, symbol, security_name, security_type, exchange)

Bond (*cusip*, face_value, coupon_rate, maturity_date)

- *cusip* refers to cusip in Security, NOT NULL.

Equity (*cusip*)

- *cusip* refers to cusip in Security, NOT NULL.

Trade (transaction_id, *account_id*, *cusip*, *counterparty_id*, trade_id, transaction_type, price, units, settlement_date, execution_date)

- *account_id* refers to account_id in Account, NOT NULL.
- *cusip* refers to cusip in Security, NOT NULL.
- *counterparty_id* refers to counterparty_id in Counterparty, NOT NULL.

Lot (lot_id, *trade_id*, *account_id*, *cusip*, units, cost_basis, acquisition_date)

- *trade_id* refers to transaction_id in Trade, NOT NULL.
- *account_id* refers to account_id in Account, NOT NULL.
- *cusip* refers to cusip in Security, NOT NULL.

Journal (journal_id, *trade_id*, *account_id*, journal_type, amount, entry_date)

- *trade_id* refers to transaction_id in Trade.
- *account_id* refers to account_id in Account, NOT NULL.

Position (*account_id*, *cusip*, as_of_date, units)

- *account_id* refers to account_id in Account, NOT NULL.
- *cusip* refers to cusip in Security, NOT NULL.

CorporateAction (action_id, *cusip*, action_type, ex_date, record_date, pay_date, adjustment_factor, new_security_name, new_symbol)

- *cusip* refers to cusip in Security, NOT NULL.

MarketData (*cusip*, date, open_price, close_price, daily_high, daily_low, volume)

- *cusip* refers to cusip in Security, NOT NULL.