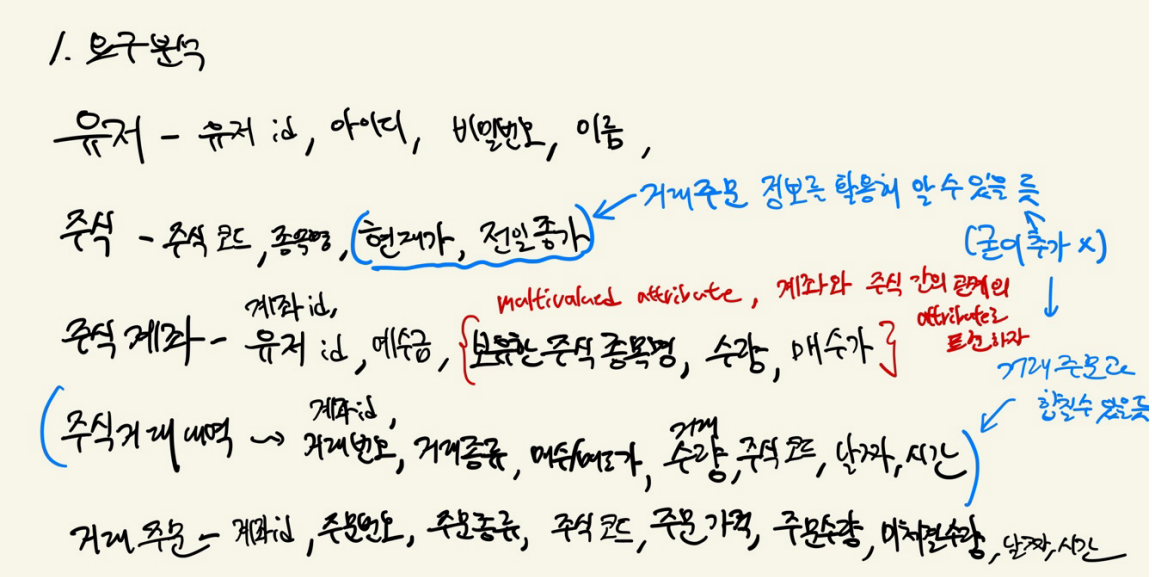
Assignment 2 설계보고서

2020081958 송재휘

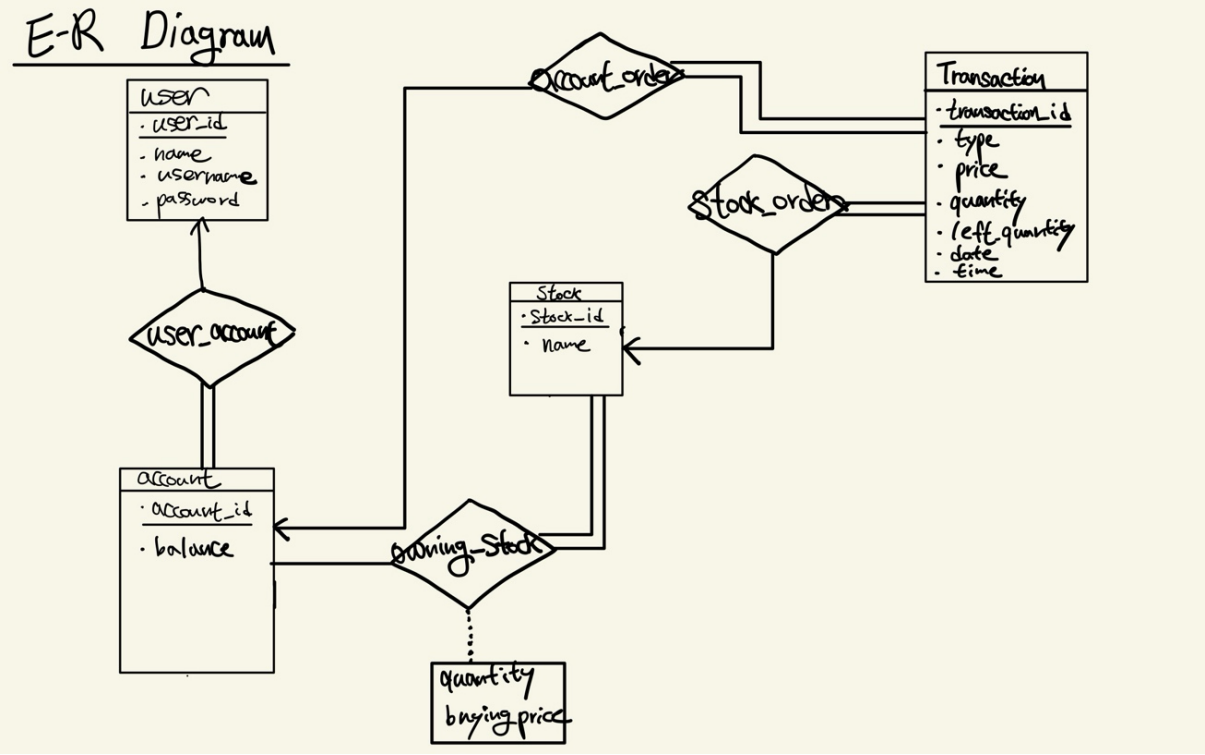
1. 요구분석



본 설계 보고서에서는 제가 그린 그림과 글을 제공하고 이에 대한 설명을 하는 식으로 내용을 구성해보고자 합니다.

위의 그림은 제가 이번 assignment2에서 간단한 HTS을 구현 하는데 필요한 요구분석 내용입니다. 저는 이번 HTS 시스템을 구현하기 위해 요구분석 과정에서 유저, 주식, 주식계좌, 주식거래내역, 거래주문이라는 큰 카테고리를 설정했습니다. 먼저, 유저의 경우 해당 HTS시스템을 이용하기 위해선 당연히 회원가입을 수행 해야 하고, 이때 만들어지는 회원 정보를 저장하기 위해 유저 식별을 위한 유저 id, 아이디, 비밀번호, 이름의 정보가 필요하다고 분석했습니다. 두번째로 주식 거래를 위해서는 주식 정보를 관리할 필요가 있다고 생각했습니다. 따라서 주식의 경우, 해당 주식을 표현하기 위한 주식코드, 종목명, 현재가, 전일종가와 같은 정보가 필요하다고 분석했습니다. 세번째로 주식 거래를 위해선 주식 계좌의 정보를 관리해야 했습니다. 주식 계좌의 경우, 저는 한 유저가 여러 개의 계좌를 만들 수 있다는 전제 하에 이를 인식하기 위해 계좌의 id와 소유주를 나타낼 유저 id, 예수금, 보유한 주식 종목명, 수량, 매수가 등의 정보가 필요하겠다고 분석했습니다. 네번째로는 거래 완료된 주식 거래 내역을 저장하기 위한 정보입니다. 여기엔 해당 거래를 한 계좌의 id, 거래 정보를 찾기 위한 거래번호, 거래 종류(매수/매도), 매수/매도가, 거래 수량, 주식코드, 날짜, 시간 등의 정보가 거래내역 관리에 필요하다고 분석했습니다. 마지막으로, 거래주문의 경우는 주식 거래가 완료되기 전에 주문 상태의 정보를 저장하기 위해 해당 주문을 만든 계좌 id, 주문번호, 주문종류(매수/매도), 주식 코드, 주문 가격, 주문 수량, 미체결 수량, 날짜, 시간 등의 정보가 필요하다고 생각했습니다.

2. E-R Diagram

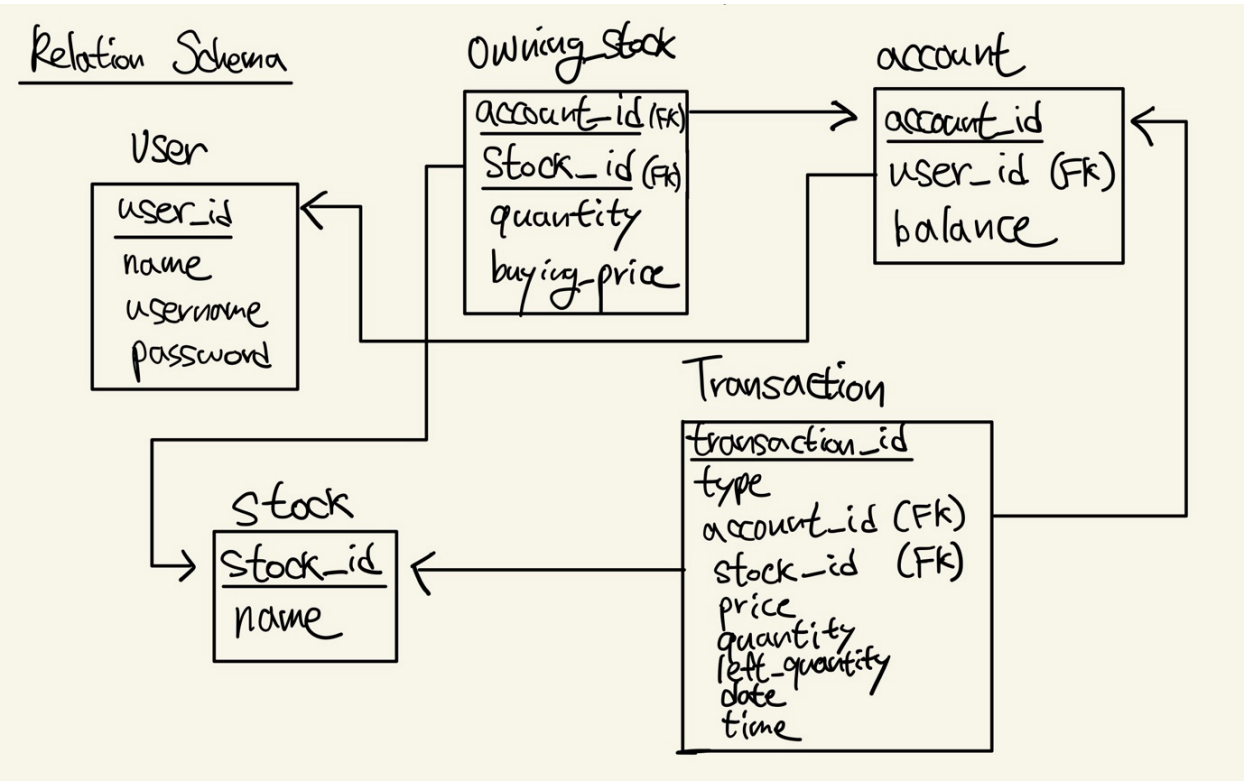


요구분석은 이렇게 완료 했고, 이를 바탕으로 ER diagram을 그리기 위해 요구분석 파트에서 추가적으로 주의 깊게 볼 부분은 파란색과 빨간색 글씨로 쓴 부분입니다. 우선 주식 거래가 완료됐을 때 정보를 관리하기 위한 주식거래내역과 거래주문의 경우 거래 주문의 미체결수량이 0이라면, 이는 완료된 거래 내역을 의미하기 때문에 개별적인 카테고리로 유지하기보단 거래주문이라는 entity set 하나만 이용해 구현해도 충분하겠다는 결론을 내렸습니다. 또한 주식 부분의 현재가와 전일종가의 경우 거래주문정보를 바탕으로 충분히 도출할 수 있기 때문에 ER diagram을 그릴 때에는 주식이라는 Entity set의 attribute에서 제외할 계획입니다. 마지막으로, 빨간색으로 표시한 주식계좌 entity set에서 보유한 주식 종목명, 수량, 매수가의 경우는 한 계좌에 여러 개의 값이 존재할 수 있는 multivalued attribute이고, 이는 주식과 계좌 사이의 관계로 볼 수 있기 때문에 relationship set의 형태로 따로 관리하자는 결론을 내렸습니다.

이를 바탕으로 그린 ER diagram은 위와 같습니다. 위의 요구분석 내용을 바탕으로 각각의 카테고리를 entity set으로 정의했고, 각 카테고리에서 필요한 정보를 해당 entity set의 attribute로 구현했습니다. 우선 User entity set의 경우는 요구분석에서 필요한 내용이 attribute로 들어가 있고 account라는 entity set과 유저의 계좌라는 의미의 user_account 관계를 맺습니다. 이때 한 유저는 여러 개의 계좌를 가질 수 있음으로 one-to-many의 cardinality를 갖도록 설계했습니다. 또한 모든 account는 user정보가 있어야 하기 때문에 total relationship을 갖습니다. Account entity set의 경우 해당 계좌를 식별하기 위한

account_id와 예수금을 나타내는 balance를 attribute로 가지고 있습니다. 이때, 유저는 주식 계좌를 통해 주식 거래를 하기 때문에 거래를 관리하는 transaction과 계좌를 통한 주식 거래라는 account_order relationship set을 갖도록 설계했습니다. 이 경우에도, 주식 계좌는 여러 거래를 할 수 있지만 한 거래는 한 주식계좌에서만 이루어져야 하기 때문에 one-to-many의 cardinality를 갖습니다. 이때 거래는 계좌가 있어야만 가능하기 때문에 total relationship을 갖습니다. 추가적으로, account entity set의 경우 주식을 나타내는 stock entity set과 보유한 주식이라는 owning_stock의 relationship set을 갖는다고 생각해 위와 같은 방식으로 설계했습니다. 이때 유저 계좌에 특정 주식을 보유 중이라면 해당 주식의 수량과 매수가 정보가 필요하기 때문에 해당 relationship set의 attribute로 갖도록 했습니다. 또한 유저 계좌에는 여러 주식을 보유할 수 있고, 한 주식도 여러 유저 계좌에서 보유될 수 있기 때문에 many-to-many로 설계했습니다. 여기서도 주식이 소유가 되려면 반드시 account 정보를 가지고 있어야 하기 때문에 total relationship을 갖도록 설계했습니다. Stock entity set의 경우 주식을 표현하기 위해 주식 코드를 나타내기 위한 stock_id와 종목명인 name이라는 attribute를 추가했습니다. 위의 요구 분석의 마지막 과정에서 언급한대로 현재가와 전일종가의 경우는 이후 transaction 내역을 통해 처리가 가능하기 때문에 ER diagram의 entity set에서 제외했습니다. Stock entity set의 경우 앞에서 언급한 relationship set 이외에도 transaction entity set과 주식 거래라는 stock_order의 relationship set을 갖는데 이때 한 주식은 여러 개의 transaction을 가질 수 있고, 한 거래는 한 주식에 대해서만 할 수 있기 때문에 one-to-many의 cardinality를 갖도록 설계했습니다. 이것 또한 거래는 반드시 주식 정보를 포함해야 하기 때문에 total relationship을 갖습니다. 이때 각각의 entity set을 식별하기 위해 고유한 id 정보를 활용해 Primary Key로 설정했습니다.

3. Relation Schema



마지막으로 설명 드릴 부분은 Relation Schema입니다. Relation Schema의 경우, 기본 원칙이 ER diagram의 각각의 entity set과 relationship set을 하나의 relation schema로 구현하는 것입니다.

우선 entity set부터 relation schema로 변환하기 시작했습니다. 앞서 ER diagram에서 그린 entity set은 User, Stock, Account, Transaction이었습니다. 이를 relation schema로 변환하는 과정에서 생긴 차이점에 대해 하나씩 얘기해보고자 합니다. 우선 User와 Stock의 경우 ER diagram에서와 차이가 없습니다. 세번째 Account의 경우 기존 ER diagram에서 user entity set과 total relationship의 one-to-many cardinality를 가졌었기 때문에 many 쪽이었던 account에 User의 primary key인 user_id를 foreign key로 추가했습니다. 마지막으로, Transaction의 경우 Account, Stock entity set 각각과 total relationship의 one-to-many cardinality를 가졌기 때문에 many 쪽에 해당한 Transaction에 Account와 Stock의 primary key인 account_id와 stock_id를 Transaction에 foreign key로 추가했습니다.

두번째는 relationship set 차례입니다. Relationship set의 경우 one-to-many cardinality 아래에서 many 쪽에 one 쪽의 primary key를 foreign key의 형태로 삽입함으로써 relationship set의 생략이 가능하기 때문에, user_account, account_order, stock_order의 관계를 생략했습니다. 이후 남은 relationship set은 owning_stock하나인데 이경우는 해당

relationship set에 참여하는 Account와 Stock의 primary key를 가져오도록 해 별도의 relation schema로 구현했습니다. 이렇게 생성된 owning_stock relation schema에는 기존에 해당 relationship set이 가졌던 attribute인 quantity와 buying_price를 포함해 Account entity set의 PK인 account_id, Stock entity set의 PK인 stock_id로 relation schema가 구성됩니다. 이때 앞서 언급한 FK의 참조 관계는 위의 그림에서 화살표로 표시했습니다.