MIS 6326 Data Management

Spring 2016

Instructor: Kevin R. Crook

Data Architecture Project

Last Name: Dalal
First Name: Shobhit
Middle Name: Divyesh

Comet Creed - *"As a Comet, I pledge honesty, integrity, and service in all that I do."*
Electronic Signature: Shobhit D Dalal

Oil Trading System

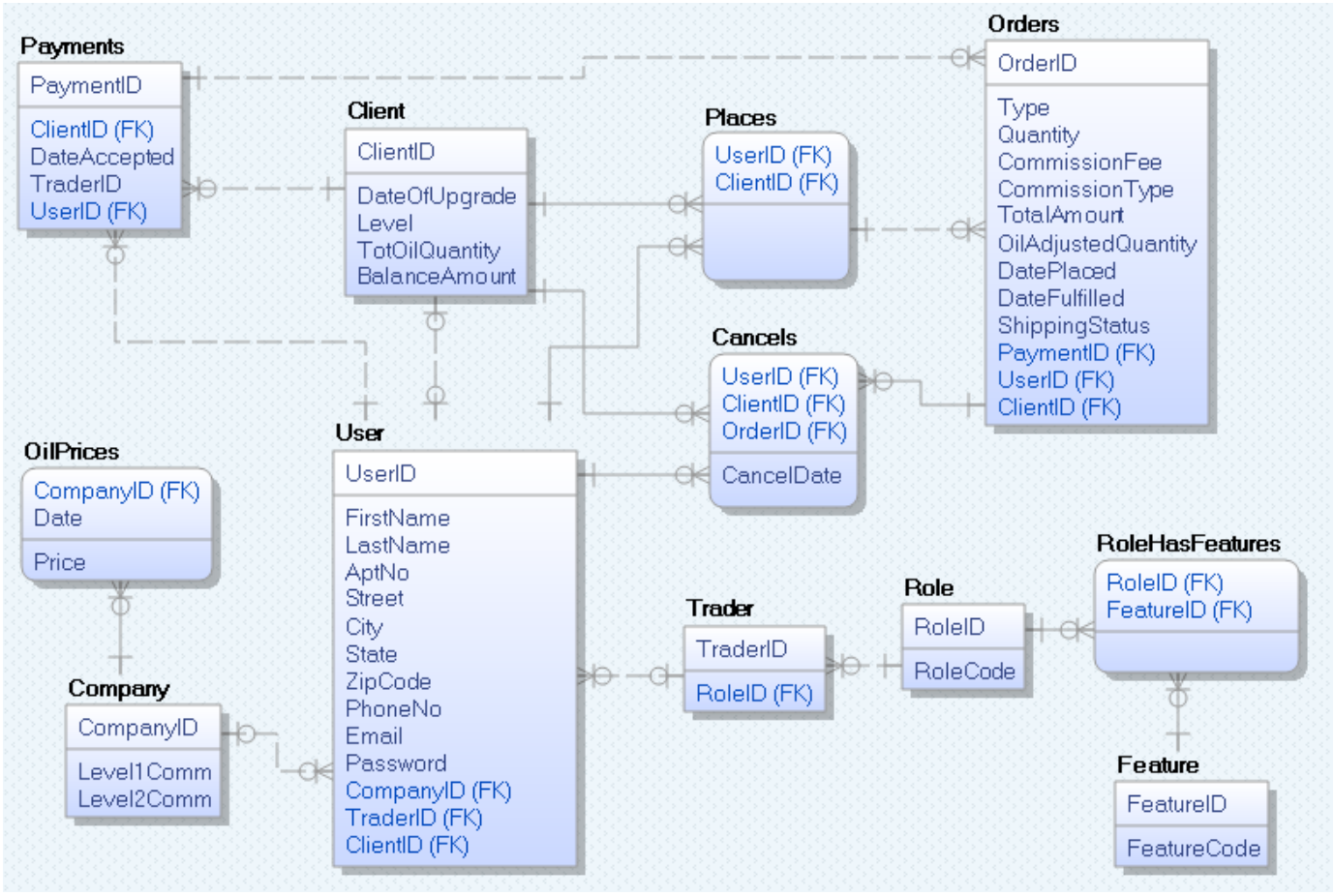
Executive Summary

Motivated by the recent increases in oil prices, a start-up idea is thought upon to set up a local shop that can sell oil to investors in Dallas. Part of the idea is to develop a software to assist traders working in her company in managing oil transactions issued by customers. In particular, end product is to create a convenient and easy-to-use software for oil traders who are trying to buy and sell oil for their clients. To help build this software, I offer to develop a database system called OTS (Oil Trading System) that is based on a relational Database Management System.

Description of database requirements:

- Each client has a unique client id generated by the system, a name (first and last), a phone number, a cell-phone number, an e-mail address, and an address (including street address, city, state, and zip code).
- Since sometimes purchased oil could be shipped to clients, it is important to retrieve the city and zip code information for each client easily.
- Each client is assigned to one of two different levels based on his or her past transaction volume. Once a client trades more than 30 barrel in any single month, the client is classified as a “Gold” customer and is charged a different commission rate for all subsequent transactions. Otherwise, the client is classified as “Silver”.
- The client also needs to specify whether he or she wants to pay the commission for the transaction in oil or cash. If the transaction fee is paid in oil, the system automatically adjusts the amount of oil left in the customer account. On the other hand, if the customer chooses to pay the commission in cash, the system must automatically compute the fee based on current oil prices.
- The value of the transaction, the date of the transaction, and the commission paid should be stored separately for each transaction.
- From time to time, clients will pay money to settle their transaction costs. For each payment transaction, you need to store the amount paid, the date, and the information related to the trader who accepted the payment.
- In some cases, traders may want to cancel certain payment and oil transactions. Although the system should allow such cancellations, logs should be stored for such cancelations for auditing purposes.

Database Entity-Relationship Diagram (ERD)



Database Design Descision Explanations

Brief Description

I have designed an Oil Trading System database which can serve three types users - client, trader, and manager. A client can directly buy/sell oil depending on his current oil reserves, or contact a trader to perform these transactions for him. Upon each transaction client pays commission to the company in form of either oil or cash. A client has a level which is initially SILVER and changes to GOLD after 30 barrels are purchased in a single month. A client's commission is applied according to his level and his oil reserves or account balance is adjusted accordingly.
The company database keeps records for each transactions. Client can pay money to the company from time to time to settle their transactional balances. A Trader can cancel certain payments and oil transactions, but these cancellations are logged for audit purposes. Payments and cancellations must go through traders only.

Database Design Decisions

- Based on the description of the OTS provided in the questions, system clearly has got 3 types of users - Manager, Trader and Client (a.k.a customer). Requirement mandates that the address information must be stored for each client at detailed level. However, we do need this information for managers as well as traders, hence this information was abstracted into a table called "User" and ISA relationship was extracted. It would be only valid to assume that a Manager is also a trader with higher privileges. Hence User can be a Trader or a Client or a Trader with higher privileges.
- Role of the trader was identified using RoleID field that has been assigned to the Trader.
- Every trader has to have a role - either Manager or Trader role. Hence Full participation is applicable for Trader Has Role relationship. Also, a Trader can have exactly single role.
- There are several requirements that are tied to the role "Capability of settling payment for orders", "Cancelling orders","Viewing reports", "Creating a Trader/Client/Manager" user, etc. In order to add support for them in scalable way, "Features" entity was introduced that was tied to a role through "Role has Features" relationship. Every feature has to belong to some role, hence full participation.
- Any user can place order but the order must be for a client. Hence ClientID has been introduced as the attribute for "User places order"
- Only trader cancels the order hence, cancels table has been introduced along with CancelDate as the attribute of the table
- A payment is accepted for selected orders is a one to many relationship between payment and orders
- Every order has to have exactly one entry in places relationship, it cannot belong to multiple entries
- Orders table was identified for holding each of the oil transaction that has been made for the client. This way we are storing commission, quantity, total amount, oil for each transaction separately. Also, ShippingStatus and DateFulfilled columns were identified for storing shipping information of the oil.
- Client is allowed to pay for one or more orders, hence one-to-many relationship was identified between "Payment Belongs to Order" relationship and Payment entity.

- Level of client, along with timestamp at which the upgrade happened has been associated with the Client entity itself. Also, we need to keep track of balance as well, hence balance oil and amount has been associated with client entity.
- Every user has to belong to exactly one company, hence the same constraint has been added in the ER diagram.
- Commissions to be applied to user levels is the information that must be maintained with company, hence they are the attributes in Company entity
- OilPrices is an entity that has been associated with the Company entity. Price maintained in OilPrices table can vary for each company (after applying state taxes, city taxes, etc) hence Prices are tied to Company through one-to-many relationship.
- "RoleHasFeatures" is a many-to-many relationship hence has been converted into a separate table.
- OilPrices table will have CompanyID and Date as the composite primary key as prices will be different for different companies as well as different for same companies on different dates
- A trigger can be defined for upgrading user's level to GOLD on completion of their 30 orders in any month