Economic Planning and Optimization of a Refinery Crude Distillation Unit
Graduate Project

By
Charles A. Pack
4/27/98

Committee Members

Dr. Stephen Dannelly
Chairman

Dr. David R. Thomas
Member

Dr. Holly Patterson
Member
# Table of Contents

1. Abstract ........................................... 3

2. Introduction ..................................... 4

3. Narrative .......................................... 9

4. Environment ...................................... 17

5. Procedure ......................................... 18

6. Results and Conclusions ......................... 41

7. Bibliography ...................................... 45

8. Appendices
   - CROP Process Overviews
     - Purchasing and Scheduling Economic Model 47
     - Process Overview 48
     - Business Functional Decomposition 49
     - Entity/Relationship Diagram 50
     - Software Architecture 51
     - Process Flow Detail 52
   - CROP Source Code
     - Database Source Code 53
       - Control Language, User Create Scripts (DCL) 53
       - Definition Language, Build Procedures (DDL) 55
       - Procedures, Functions, Triggers (PL/SQL) 64
       - Demo Data Scripts and Select Queries (DML) 89
       - Referenced Source Code 96
     - User Interface (Java) Source Code 103
   - Test Data Validation, Results 168
   - Refining Background Information
     - Simplified Refinery Process Overview 169
     - Crude Oil Distillation, Temperatures, & Products 170
     - Glossary of Terms 171
Abstract

This project is the design and implementation of a prototype economic planning and scheduling optimization system for a petroleum refinery crude distillation unit (CDU). The project incorporates economic principles, optimization models, organic chemistry fundamentals, crude oil refining principles, engineering constraints of a refinery crude unit, mathematical algorithms, and a relational database management system.
Introduction

Petroleum refiners determine what products they will make, with the goal of maximizing profits, based on the supply and cost of various types of crude oils, the physical constraints of the refinery, and the forecasted demands of the market. Optimizing the selection and scheduling of the inputs to the crude oil distillation unit (CDU) is the first step in overall refinery process optimization.

Crude oil is composed of a variety of hydrocarbons ranging from methane, the lightest, up to extremely heavy asphaltenes. Crude oils come from all over the world, and the composition of any two crudes can vary greatly. Some contain higher amounts of sulfur, nickel, or other undesirable elements; some contain greater amounts of the heavy hydrocarbons, others contain greater amounts of the lighter hydrocarbons. The crude oil known as "West Texas Sour" has a completely different chemical composition than that of "Saudi Arabian light". Refineries are designed and built to utilize crudes that have specific characteristics and are in a specific composition range. Refiners will purchase shipments of crude oil within a specific composition range, and forecast profits based on the types and amounts of products they will make.

A typical refinery is segmented into processing "units" - each takes as input specific compounds or feedstocks, and using a process particular to that unit, produces as output a different group of compounds.¹ For example, the CDU takes crude oil as its feedstock, and produces (ordered lightest to heaviest) Liquified Petroleum Gas (or LPG, Butanes and lighter

¹ See appendix, "Simplified Refinery Process Overview"
gasses), Straight Run Gasoline, Naphtha, Kerosene, Light Gas Oil, Heavy Gas Oil, and Straight Run Residue. The CDU requires specialized equipment and is optimized for separating these particular hydrocarbons from the crude oil. The Catalytic Cracker, or "Cat Cracker" receives Heavy Gas Oil from the CDU, and produces LPG, Gasoline, Light Gas Oil, Heavy Gas Oil, Cycle Oil, and Coke using a distinctly different set of processes than the CDU. Because the Cat Cracker's primary purpose is to convert Heavy Gas Oil into LPG and Gasoline, it requires a different set of hardware than the CDU. Each unit within a refinery is optimized to make specific products from specific feedstocks, therefore each has different physical equipment and each has a unique economic model. A typical refinery may have thirty or more processing units. Each unit has a physical constraint on feedstocks and outputs, or throughput; since the majority of feedstocks are processed by the CDU first, it is the most constraining unit in the system. Downstream units are designed to handle fluctuations in outputs from the CDU, and will produce volume based on projected market demands. Ideally, the CDU runs as close to operational capacity as possible, and attempts to produce the most profitable outputs.

Because each refinery is designed and built with the market for its products in mind, each has a specific range of products and volumes that it can produce. Each refinery strives to maximize profits based on sales volumes of their products. If the CDU produces too much of one range of outputs for which there is no economic incentive, then the contribution to overall refinery profitability is reduced. If the CDU produces intermediate products (intermediates) that could have been used for more profitable end-products, then the refinery has missed the opportunity to make more money.

---

2 see appendix, "Crude Oil Distillation, Temperatures, & Products"
The planning phase begins with a series of contracted sales requirements, and forecasted market conditions such as seasonal heating oil, asphalt, and gasoline demand. The refinery economist forecasts which products will bring the greatest return, in what volumes, and if the refinery can meet the demand. Crude composition determines products, so the purchaser must base crude selection on market demands. The current preferred method for determining which crudes to purchase is a linear programming tool (LP), which optimizes selection based on profit maximization or cost minimization. The LP considers the physical constraints of the refinery, the physical composition and characteristics of crudes (crude assay), availability and price of the crudes, and the price of each product. The purchaser will then use the short list of preferred crudes to select which ones to purchase. In a more sophisticated system, buyers, traders, marketers, planners, schedulers, economists, and economic engineers are involved in the decision process. Also, in some situations a crude purchase may come down to whatever is available on the market, regardless of cost and characteristics.

The planners or schedulers must then optimize the scheduling of the crude shipments to and within the refinery. They will also schedule crude "blending" in order to ensure that the most optimal crude mix is sent to the CDU. Optimal blending and scheduling will contribute to maximum CDU throughput, and minimize the swing in "modes of operation". Blending and scheduling can both be solved with an LP tool. The output of the CDU will determine quantities available for downstream units, which will impact product profitability.
Daily, U.S. refineries process approximately 15 to 16 million barrels of crude oil, with an estimated gross margin of $2 to $7 per barrel.\textsuperscript{3} With a national margin ranging from $30 to $112 million daily, one can understand the importance of optimization. These market demands are seasonal, but continuing upward.

The CDU was selected for this project because it is the initial step in a refining process, the value and composition of the feedstocks are definable, and the quantity and price of products can be determined fairly accurately. The CDU can also be viewed as the first producer in a refinery internal market. Each downstream unit is dependent on the products from the CDU, or must purchase feedstocks from the outside market. Once the CDU is modeled and optimized, we can investigate each downstream unit separately.

In summary, refiners can make more money if they optimize production, based on optimal crude oil selection and scheduling. This project demonstrates a way to assist the refiner, by prototyping a tool that assists the user in optimizing the inputs to the CDU.

**Project Objective**

The objective of this project is to match the business model for economics and planning, and integrate the following business functions into a single, cohesive, and easy-to-use software package: select and purchase optimal crude oils, schedule shipment and storage, schedule product sales to customers, and determine the dollar-value of the crude-oil purchase. This software prototype, named "CROP" for CRude OPtimizer, models each of the above mentioned

\textsuperscript{3} Oil & Gas Journal, August, 1998. Refining is near capacity in the U.S. Gross margin numbers are affected by market demands, seasons, and politics.
business functions. It does not provide a complete solution, but demonstrates the ability to integrate the business functions into a cohesive software package.

**Justification for the Project**

The current refining industry approach is to use best-in-class software for each of the required functions mentioned above, and then attempt to "integrate" the output from each system into valuable information. Attempts have been made to use a data-warehouse approach to integrate the data: the effort is costly and time-consuming, and it has not yet been determined if this is a viable solution. This project attempts to show a path that will reduce the number of software interfaces required, reduce the complexity of the client computer environment, centralize the data required to make economic decisions, and reduce the administrative overhead required to support the diverse products that the end-user must utilize.

Another trend in refining & process industry software is for a company that has a successful product (an LP, for example) to purchase another company that is a loss-leader or non-competitive developer of another niche software package, then attempt to provide a modular approach to the customer's needs. This can be seen as an attempt to increase market share, but has not resulted in the production of truly "integrated" solutions similar to the what the "CROP" prototype attempts to prove.
**Narrative**

The CRude OPtimizer software (CROP) helps the refinery crude-oil purchaser, economist, planner, and scheduler make selection and scheduling decisions by accomplishing the following:

- Allows a person to select optimal crudes based on characteristics, cost, and product requirements.
- Allows a person to indicate when a crude is purchased and scheduled for delivery.
- Allows a person to schedule crude oil flows into and products out of the CDU.
- Allows a person to compare the Gross Margin of the crude selections.

The CROP software does so by considering the following information:

- characteristics of crudes and products,
- cost of crudes and price of products,
- refinery throughput constraints,
- crude oil delivery and product shipment schedules,
- profit maximization and cost minimization goal (profit maximization).

The CROP software consists of:

- A linear programming model, designed to optimize the selection of crude-oils purchased by a refinery.
- A model for scheduling crude-oil movements into and products out of the CDU.
- A mechanism for determining the projected Gross Margin of the products from the CDU after distillation.

**Components of the CROP Software**

The CROP software is segregated into conceptual modules, which can work independently, but appear as a tightly-integrated desktop application. CROP consists of a Refinery Database module (RD), a Linear Programming module (LP), a Shipment and Delivery module (SD), a Scheduling module (SCHED), and a Gross Margin Determination module (GMD). Each module is designed to interface with the other, and each is composed of database...
objects and client applets. The CROP client software is designed to be platform independent, and the server components use an Oracle relational database to store data.

**Refinery Database module**

The RD module consists of a user interface (UI) client front-end and a relational database (RDB) server back-end. The UI allows the end-user to enter configuration data directly into the RDB via a client applet. The RDB, which resides on a server computer and, stores information about the refinery, crude pricing, crude assays, product pricing, shipment and delivery scheduling, inventories, and LP constraints. The end-user updates configuration data that is stored in the RDB via the UI, which is a set of Java applets designed to interface with specific components of the RDB.

For example, when the assay changes for a specific crude oil, the end-user modifies the record for that crude oil assay in the RDB by using the crude-oil assay data-entry applet. The RDB stores the results of each of the other modules responsible for performing calculations in the CROP software.

**Linear Programming module**

The Linear Programming (LP) module uses crude oil and product data from the RDB to recommend which crudes should be purchased. The end-user selects a set of available crudes from the RDB using the UI, and the LP module compares the cost of the crudes and price of the products, component ratios of the crudes required to manufacture the products, and product and crude volume constraints. The LP module determines the profit-maximizing combination of crudes to purchase and product volumes to produce. The results of an LP calculation, or
"optimization run" are written to the RDB for end-user viewing and reporting. The UI contains applets to view the projected product mix and profitability, based on the optimal crudes to purchase.

**Shipment and Delivery module**

The Shipment and Delivery (SD) module allows an end-user to log a crude-oil purchase in the RDB and schedule its delivery to the refinery. Through the UI, the end-user indicates which crude oils the refinery wishes to purchase (based on optimal crudes recommended by the LP), and when the crude will be delivered. The end-user also uses the SD module to log a sales shipment in the RDB, which indicates that a product has been removed from the refinery storage. The SD module is used to indicate a purchase following an LP comparison of optimal crudes: once the crudes have been recommended and stored in the RDB, the end-user selects which crudes to actually purchase.

**Scheduling module**

The Scheduling module (SCHED) is used to schedule crudes from the refinery tanks to the CDU, and product movements from the CDU to product tanks. Crude shipments logged in the RDB by the SD module become part of the refinery inventory, with processing at the CDU as the final step before pumping to product tanks. Current and predicted tank volumes for crudes and products are stored in the RDB; intra-refinery crude and product movements occur through the SCHED module, and are stored in the RDB. The "Crude Run" applet, a component of the SCHED module, simulates the production of CDU products based on the crude assay of the scheduled crudes. When a "crude run" occurs for a given day, crudes from the crude schedule
are fractioned into the mix of products, and the results are stored in the product schedule table. The end-user can then schedule the product deliveries through the "product schedule" applet.

**Gross Margin Determination module**

The GMD module is used to determine the effectiveness of the purchase and operational decisions. Each purchase, sale, and "Crude Run" is stored as a separate row in the RDB, so that the profit and characteristics may be compared to any other. Also, any scheduling changes that may occur can be compared to forecasted outputs, with a cost/profit differential indicated. The GMD requires no data input from the end-user; it is for viewing only.

**Steps in the CROP process.**

**Software Installation**

The client software runs on any computer that supports the Java Virtual Machine architecture. The client software is distributed through the world-wide-web, via a download. The client computer hardware is irrelevant; the only constraint is that the client computer can run a Java application. The server components require an Oracle database instance. See appendix for database build procedures. Once the database is installed and accessible from the client computer via JDBC/ODBC, the client is ready to begin working.

**Running CROP**

The CROP software is designed to be platform-independent: that is, any machine that can run a Java application can run CROP. The CROP software looks just like any other application on the computer desktop. Depending on the client operating environment, a double-click on the
CROP icon will launch the application. Once launched, the CROP application requires a user login ID and password to gain access to the database.

**Base Configuration**

Once the software is installed and running on the client and the server database is created, the user configures the database for the specific refinery, using data entry screens provided by the UI module (for large-scale data loading, SQL scripts can be executed from an SQL query tool, such as Oracle's SQL Plus or SQL*Loader—see appendix for demo data). For example, to enter information about potential crudes, the end-user launches the CROP application/applet, selects the "Crude Oils" menu item, and begins entering data in the data grid.

**Refinery Configuration**

- Configure the CDU process unit(s) in the Unit table.
- Enter the Crude and Product Tanks into the Tank table.
- Enter the Stream connections between the Tanks and the Units.

**Crude Oil Configuration**

- Enter Crude Oil names and basic info into the Crude table.
- Enter Initial Crude Assay data into the Crude Assay table.
- Enter Available Crudes, based on what is available on the market.

**Product Configuration**

- Enter the product names, prices, and descriptions in the Product table.
Crude Selection

Once the RDB is configured for the refinery, products, and potential crudes, the user selects crudes and compare them by using the LP module. The end-user accesses the main application, selects the "Available Crudes module" button, and selects available crudes that may be purchased. Once crudes are selected (and LP constraints entered), the end-user enters a date and description, then presses a button in the LP module to execute the LP calculations.

Select Crudes to Compare

- Configure the crude assay – Is the current Assay correct?
- Configure the price of the crude (Shipment Price/Barrel).
- Enter the volume of the available crude.

Select Optimal Crudes

The user selects crudes for purchase based on the results of the LP. The LP recommends which crudes to purchase, but the user has the final choice. The end-user selects a crude for purchase, and assigns a preliminary shipment date using the "Shipment and Delivery module". The crude oil purchase is then immediately available to the Scheduler in the "Crude Scheduling module".
**Scheduling - Crudes**

Once a crude shipment is purchased, the user schedules its flow to and within the refinery. The end-user accesses the application, selects the "Scheduling module", "Crude Schedule" command button, and begins evaluating the contents of the crude schedule data grid. To assist the end-user, a refinery flow simulator is available from the "Scheduling module" menu. The flow simulator shows maximum throughput from the crudes in the crude schedule to the crude units based on tank, stream, and CDU capacities.

**Schedule Crudes to the Refinery**

- Crude shipment – when will it be shipped, and when will it arrive?

**Schedule Crudes within the Refinery**

- Schedule the shipment to be stored in a tank or tanks.
- Determine when it will be pumped to the CDU?

**Scheduling - Products**

Once a crude run occurs, the product schedule contains a new batch of products to schedule to process and storage tanks, and out of the refinery. The end-user accesses the application, selects the "Scheduling module", "Product Schedule" command button, and begins evaluating the contents of the product schedule data grid.

**Schedule Products**

- Product shipment – where will we store it, and when will it be shipped out?
- Product Sale - ship a product to a customer.
Gross Margin Determination

Once a crude shipment has been processed into products, we need to determine if its purchase and distillation were the most profitable choice we could have made. To launch the GMD module, the end-user launches the application, selects the "Gross Margin Determination module" command button, and views crude purchases to evaluate.

Compare Cost of Crude to Price of Products

- Select a scheduled crude run (sort by batch ID, or by date).
- Compare the cost, revenue, and profit.
- GMD info is stored in RDB, so we can compare crude runs over time.
Narrative - Summary

The CROP software was designed so that comparing crudes with the LP, selecting crudes for purchase, scheduling crudes into and products out of the CDU, and determining if the purchase was a good decision will each be a simple process. The CROP software helps simplify the overall process by integrating previously separated tasks. The quick decision turnaround – using the LP, SD, SCHED, and GMD modules helps the end user determine if a purchase is a good decision before the crude is actually run through the refinery. The CROP software assists the purchaser, economist, scheduler, and planner in their decision-making process.

---

4 see appendix, "CROP - Process Overview"
5 see appendix, "Purchasing and Scheduling Economic Model"
Environment

Client Configuration

It is the intent of the designer to make the client side of the CROP software as platform-independent as possible. The most widely used platform available is the world-wide-web browser, and the Java language is the most widely used www language; therefore, the client piece of the software is based on Java technology. The only constraint is the client computer's ability to execute a Java-enabled application or applet. The client software for CROP was tested on personal computers with the following configuration:

- IBM compatible - Intel Pentium-based personal computer,
- Microsoft Windows 95 operating system.

Server Configuration

While the CROP software is designed with platform independence in mind, the following server configuration is the primary test ground:

- Oracle relational database management system (DBMS),
- Microsoft Windows NT Server operating system,
- Microsoft Internet Information Server (IIS) web server.

All data access by CROP is handled by the Oracle DBMS built-in security.

Programming Languages used:

- Java (JBuilder™ Professional v.1.0), SQL, PL/SQL.
**Procedure**

The CROP software utilizes an Oracle relational database and stored database procedures for data manipulation, and Java applications for the user interface. Each of the modules of the software performs discrete but related functions. Each module can be used independently to accomplish a specific task or business function, or in series to accomplish a full purchasing and profit analysis cycle. Each module accesses the database, and delivers information through the Java client.

**Refinery Database module**

The RD module consists of a user interface (UI) and a relational database (RDB). The primary function of the UI is entry of configuration data. For example, the UI is used to enter crude oil assay information, product prices, and LP constraints. While each of the other modules has a user interface, those interfaces are dedicated to processing the configuration data. The RDB is the repository of all CROP information, and is based on the Oracle RDBMS. While the intent is to keep the database design as open as possible, Oracle-specific database procedures are utilized whenever optimal.

The RDB contains:

- crude characteristics, price, availability, and preferences,
- purchase options (LP solutions),
- shipment volumes, schedules, and price,
- refinery description and constraints (throughput, storage),
- product characteristics,
- product sales prices and delivery dates,
- refinery tank inventories,
- customers.

The UI provides access to the RDB. It allows:
- data insert, update, and delete,
- search and selection.

All data used by the CROP system is stored in the RDB. Direct user input to the RDB occurs through the UI. Each of the other modules utilizes data stored in the RDB, and is also responsible for inserting data into the RDB through user-activated processes, such as: executing the Linear Program, which inserts crude purchase recommendations; and selecting a crude for purchase via the Shipment and Deliver module, which marks a crude selection as "purchased".

**Linear Programming module**

The LP component uses crude oil and product data from the RDB to recommend which crudes should be purchased. The LP is executed any time an optimal crude purchase is required. The LP compares crude characteristics to product requirements, and determines which crudes can deliver the quantities of product desired. The LP compares the cost of the crudes, and determines the ratio of crudes that should be purchased to optimize profit and meet the product quantity requirements. The LP model is based on the following data stored in the RDB:

- Variables
  - crude oils, cost,
  - products, market price.
• Constraints
  • crude oil availability, supply,
  • distillation capacity,
  • product maximum demand.
• Objectives
  • minimize cost of crudes purchased,
  • optimize distilled product ratios,
  • maximize profits.

The LP is written with PL/SQL, SQL, and Java. The application requires data extraction from the RDB based on user-selected available crudes. The application performs calculations based on the cost of the crudes, the price of the products, user-determined volume constraints, and the ratio of the components of the crude. Through the use of the simplex method for linear programming optimization, the optimal ratio of crudes that will maximize profit is determined. The actual LP algorithms are implemented with PL/SQL.

The LP module uses crude oil data as inputs, and outputs the name and recommended volume of one or more crudes to purchase. The following tables are used for input: crude, product, crude assay. The "LP constraints" table is used to bound the LP. Crude purchase recommendations are inserted into the "optimal crude" table as a "recommended" purchase, but do not become an actual shipment until a purchase date is entered for the specified crude. The LP also creates projected product volume entries in the "LP Products" table.
Shipment and Delivery module

The SD module logs a purchase, and schedules its delivery. The purchase then becomes available for the scheduler. The SD module is responsible for logging a purchase transaction (lock in a shipment), and also logging a sales delivery, which clears the product from the refinery storage. The SD uses the results of the LP to select crudes for purchase, or it can be used independently to select a crude that is available for purchase: the LP recommends, the SD selects. All SD transactions are stored in the RDB, and use the following data:

- For Crudes:
  - crude name, characteristics, cost, shipment volume, delivery date,
  - purchase agreement information (description).
- For Products:
  - product name, volume, price,
  - shipment date, description, customer

The SD module uses a combination of Java and PL/SQL. The PL/SQL components are responsible for data manipulation, calculations, and transactions: the Java applets are responsible for presenting information.

The SD module works primarily with the "optimal" and "purchased" crude table. It is responsible for updating records to indicate that a shipment is "locked in". The SD module also uses information from other tables (stream, tank, schedule) to ensure that the refinery inventory is stable.
**Scheduling module**

The SCHED module is used to schedule crudes into the refinery storage and through production to product storage. SCHED indicates gaps in the crude supply, crude inventory levels, projected product requirements, and refinery throughput changes. The Shipment and Delivery module (SD) works closely with the SCHED module. The SCHED module considers:

- tank capacities and inventories,
- simulated refinery flows,
- crudes purchased and in-transit,
- refinery throughput constraints,
- product delivery schedules.

The schedule is set for any number of days in the future, based on the current and projected crude supply. The user is presented with a schedule that shows:

- the mix of crudes scheduled to be delivered to the refinery,
- the current crude inventory,
- expected throughput,
- the current product inventory.

The SCHED module relies heavily on algorithms implemented in PL/SQL to perform load balancing, simple calculations, and data manipulation in the RDB. Java is used to present the data to the client.
The Scheduling module works primarily from the crude and product schedule, stream, unit, tank, and purchased crude tables. A schedule must take into account crude and product flow rates, refinery constraints, tank volume inventory, products, and purchase shipments and deliveries. The end-user has the ability to modify flow rates and make tank inventory adjustments. The end-user can simulate refinery flows based on current crude inventory, and simulate a "crude run" that moves crude from the crude schedule to products in the product schedule. The output of the Scheduling module is a working schedule for the refinery.

**Gross Margin Determination module**

The GMD module is used to determine the effectiveness of the purchase and operational decisions. Each purchase, sale, and "Crude Run" is stored as a separate entity, so that the profit and characteristics can be analyzed and compared to other entities in the RDB. The GMD module is a simplified income statement, with the crude oil purchases representing the expenses, and the product sales representing the revenues. The GMD module uses:

- crude oil shipment cost and volume,
- product prices and volumes for a crude run.

The GMD module requires no data manipulation, but does require reading data and delivering results to the client software, and so primarily consists of SQL queries returning data to the client-side Java applet. The Java applet allows read-only access to the GMD module. The GMD module uses the price, volume, and crude assay information from the purchased crude table, and uses the price and volume from the product table. The GMD module performs calculations on the data, but does not store the computed data in the database. The computed data is delivered to the end-user via the GMD module data grid.
CROP Frame-byFrame Description

Frame Overview

Each frame described below is resizable, movable, and independent. Only the "Main Menu" contains a menu bar across the top. Once a frame/applet is launched, the end-user can change focus to another frame/applet and launch additional frames. Within each frame, the data grid displays the data and allows column-width and row-height adjustment. Each data frame includes a status bar (at the bottom) that indicates the current row position and total row count. Each data frame also contains (at the top) a data navigation bar that allows insert, delete, commit, and basic navigation through the underlying data table. The CROP application is closed only from the "Main Menu", also referred to as "Frame 1".

CROP Database Login

Before the client software can access the database tables, the end-user must provide a database user ID and password, then press the "OK" button.
Main Menu - Frame 1

The "Main Menu" of the CROP application immediately follows user login. Every end-
user will see this frame before any of the other frames. The "Main Menu" contains the UI access
to the RDB through the Menu Bar and Menu Items at the top of the frame. The Menu Items
provides access to the Crude, Crude Assay, Tank, Unit, Stream, Product, Customer, and LP
Constraint tables.

The command buttons in the main body of Frame1 lead to the list of available crudes
(Frame 4), and each of the other modules: LP (Frame 20), SD (Frame 21), SCHED (Frame 18),
and GMD (Frame 13).
Crude Oil - Frame 2

Frame 2 is the data maintenance applet for the list of Crude oils. The database table

contains the Crude ID, or unique identifier, and the full name of the Crude. The Frame has no database procedure tied to its function.

Crude Assay - Frame 3

Frame 3 serves much the same function as Frame 2, except for the Crude Assay table instead of the Crude table. Again, no database procedures are tied to this table. The end-user can insert, update, and delete Crude assay information, with deletes allowed only if no activity has occurred in other tables for the particular assay. Data integrity constraints are enforced at the database level, so no logic is required at the client-applet level.
Available Crudes - Frame 4

Frame 4 allows end-user access to the Available Crudes Table. When inserting, the end-user selects a Crude Assay for the Available Crude, enters a unique identifier (ID), the going price for the crude (per barrel), and the total volume available. The "Selected" column is used by the LP module: the value "Y" indicates that the Available Crude is to be considered by the LP when it evaluates crudes.
**Purchased Crude - Frame 5**

Frame 5 indicates which crudes have been selected for purchase. Rows are entered into the Purchased Crude table automatically when an end-user sets the purchase column to "Y" for a row in the Optimal Crude table. When the end-user enters an initial delivery date, the crude is immediately copied to the Crude Schedule via a database trigger.

**Refinery Process Units - Frame 6**

Frame 6 allows the end-user to configure the Refinery Processing Units.
Refinery Tanks - Frame 7

Frame 7 is the data maintenance screen for the Tank table. No database procedures apply. The end-user modifies min and max volume, ID, description, and type. The Current Volume column is manually reconciled and updated each morning.

<table>
<thead>
<tr>
<th>ID</th>
<th>DESCRIPTION</th>
<th>MAX VOLUME</th>
<th>MIN VOLUME</th>
<th>CURRENT VOLUME</th>
<th>TYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>T001</td>
<td>Number One Crude Tank</td>
<td>100,000</td>
<td>1,000</td>
<td>75,000</td>
<td>CHARGE</td>
</tr>
<tr>
<td>T002</td>
<td>Number Two Crude Tank</td>
<td>15,000</td>
<td>1,000</td>
<td>12,000</td>
<td>CHARGE</td>
</tr>
<tr>
<td>T003</td>
<td>Number Three Crude Tank</td>
<td>10,000</td>
<td>500</td>
<td>8,000</td>
<td>CHARGE</td>
</tr>
<tr>
<td>TETH</td>
<td>Ethane Tank</td>
<td>10,000</td>
<td>100</td>
<td>9,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TMET</td>
<td>Methane Tank</td>
<td>10,000</td>
<td>100</td>
<td>9,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TDES</td>
<td>Diesel Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TPPN</td>
<td>Propane/Propylene mix Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TIC4</td>
<td>Iso-Butane Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TNC4</td>
<td>Butane Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TCS1</td>
<td>Straight-Run Gas Tank</td>
<td>10,000</td>
<td>100</td>
<td>9,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TLN1</td>
<td>Light Naphtha Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TMN1</td>
<td>Medium Naphtha Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>THN1</td>
<td>Heavy Naphtha Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TJet</td>
<td>Jet Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>THD8</td>
<td>Heavy Diesel Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TLOO</td>
<td>Light Gas Oil Tank</td>
<td>10,000</td>
<td>100</td>
<td>6,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TVH1</td>
<td>Heavy Vacuum Gas Oil Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TDC1</td>
<td>Resid Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TVB1</td>
<td>Vacuum Tower Bottoms Tank</td>
<td>10,000</td>
<td>100</td>
<td>6,000</td>
<td>PRODUCT</td>
</tr>
<tr>
<td>TVL6</td>
<td>Light Vacuum Gas Oil Tank</td>
<td>10,000</td>
<td>100</td>
<td>8,000</td>
<td>PRODUCT</td>
</tr>
</tbody>
</table>

Master Product List - Frame 8

Frame 8 is responsible for tracking the price for each product. Because the CDU produces a specific volume of set products for each crude, the number and type of products will not change. Since this software is a prototype of the CDU only, no special features are included.
to track multiple product slates. For this project, the list of products is static, with changes only to the price.

**Refinery Streams - Frame 9**

Frame 9 is the data entry module for refinery streams, or flows between tanks and units.

The streams determine flow capacity, and flow feasibility. The crude flow simulator uses data from the streams table to determine maximum throughput.
**Crude Schedule - Frame 10**

Frame 10 allows the end-user to modify and fill in details for the Crude schedule. The frame also has two non-editable fields that display the purchased-crude information. The "Crude Run" process will read entries in the crude schedule and create products in the product schedule.

![Crude Schedule Image]

**Product Schedule - Frame 11**

Frame 11 is mostly identical to Frame 10, except that it works on the Product Schedule, instead of the Crude Schedule. The end-user will utilize Frame 11 to schedule products to be delivered.

![Product Schedule Image]
LP Run - Frame 12

The LP Run applet allows the end-user to set up LP information before they launch the LP process. By entering and committing a row, the user will trigger the LP to run. When the LP completes, the Profit column will display the projected profit the company can expect if the user follows the LP recommendation, and purchases the optimal crudes.

GMD Module - Frame 13

The GMD module is a simple read-only frame that allows the end-user to view the results of a crude purchase, run, and product sales from the crude. If no products have been created yet from the purchased crude, then revenue and gross margin will be null.
**LP Constraints - Frame 14**

Frame 14 allows the LP manager to indicate constraints on the LP. These include CDU throughput, and crude and product maximum volumes. The LP module uses these constraints in its calculation of the recommended optimal crudes to purchase.

**Optimal Crudes - Frame 15**

The Optimal Crudes entries are created by the LP run process. To purchase an optimal crude,

update the "purchase" column, set to "Y".
LP Products - Frame 16

Frame 16 displays the product mix from each LP run. Each row indicates the volume of product that the refinery can expect to make if it follows the purchasing recommendations from the LP run. This frame is read-only.

<table>
<thead>
<tr>
<th>LP Run</th>
<th>LP Date</th>
<th>Optimal Crude ID</th>
<th>CRUDE ID</th>
<th>PRODUCT_NAME</th>
<th>VOLUME</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>4/1/99 10:30:59 PM</td>
<td>12</td>
<td>ORE</td>
<td>NC4</td>
<td>94.1</td>
</tr>
<tr>
<td>15</td>
<td>4/1/99 10:30:59 PM</td>
<td>12</td>
<td>ORE</td>
<td>PPN</td>
<td>18.61</td>
</tr>
<tr>
<td>16</td>
<td>4/1/99 10:30:59 PM</td>
<td>12</td>
<td>ORE</td>
<td>V31</td>
<td>4,071.59</td>
</tr>
<tr>
<td>17</td>
<td>4/1/99 10:30:59 PM</td>
<td>11</td>
<td>KWT</td>
<td>CB1</td>
<td>48,979.34</td>
</tr>
<tr>
<td>18</td>
<td>4/1/99 10:30:59 PM</td>
<td>11</td>
<td>KWT</td>
<td>DC1</td>
<td>31,409.50</td>
</tr>
<tr>
<td>19</td>
<td>4/1/99 10:30:59 PM</td>
<td>11</td>
<td>KWT</td>
<td>DL1</td>
<td>68,745.38</td>
</tr>
<tr>
<td>20</td>
<td>4/1/99 10:30:59 PM</td>
<td>11</td>
<td>KWT</td>
<td>ETN</td>
<td>1,177.86</td>
</tr>
<tr>
<td>21</td>
<td>4/1/99 10:30:59 PM</td>
<td>11</td>
<td>KWT</td>
<td>HD1</td>
<td>41,814.02</td>
</tr>
<tr>
<td>22</td>
<td>4/1/99 10:30:59 PM</td>
<td>11</td>
<td>KWT</td>
<td>HN1</td>
<td>24,148.13</td>
</tr>
</tbody>
</table>

Scheduling Module - Frame 18

This frame serves as a sub-menu to the SCHED frames. The "Run Crudes" button will simulate a crude run and create entries in the product schedule table, based on entries in the crude schedule table; "Simulate Crude Schedule Flows" will populate the Simulated Flows table.
Crude Run - Frame 19

Frame 19 allows the end-user to start a crude run, by entering in the schedule date and the crude unit. Commiting the change will fire the database procedure to simulate the crude run.

LP Module - Frame 20

Frame 20 is the user interface to the LP module. The "Run LP" command button will launch the LP module applet, which allows the end-user to enter the starting information for the LP run. "Optimal Crude Results" launches the applet that displays optimal crudes, and allows the end-user to select crudes for purchase. The "Product Mix Results" displays the table of products that will be manufactured if the LP recommendations are followed.
Shipmenet and Delivery Module - Frame 21

Frame 21 is the menu for the SD - Crude Purchasing (Shipment & Delivery) module. This applet allows the end user to access the optimal crudes (and purchase them), and view purchased crudes, that are ready to schedule.

Simulation Run Frame - 22

This frame allows the end-user to simulate the flow of all crude oil in the crude schedule table, to and from tanks, and to the crude units. This serves as an inventory simulation, with the intent to maximize flows for each stream, unit, and tank. The end-user can then view the results in the simulated flows frame, and determine how to schedule the crudes.
Simulated Flow - Frame 23

Displays the results of the "Simulation Run" against the Crude Schedule table. This

<table>
<thead>
<tr>
<th>BRT_ID</th>
<th>SIM_DATE</th>
<th>FLOW_VOL</th>
<th>SCHEDULE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3/1/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3/2/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>3/3/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>3/4/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>3/5/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>3/6/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3/7/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3/8/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3/9/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3/10/98 5:18:45 PM</td>
<td>100,000</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>3/11/98 5:18:45 PM</td>
<td>50,000</td>
<td></td>
</tr>
</tbody>
</table>

Record 1 of 155

frame is read-only, and is displayed along with the "simulation run" frame.

Customer List - Frame 24

The list of product customers, accessed from the main menu: this table is used along with the product schedule and product sales tables.

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Joe Smith Co. Gas Stations</td>
</tr>
<tr>
<td>2</td>
<td>PEMEX</td>
</tr>
<tr>
<td>3</td>
<td>Houston Intercontinental Airport</td>
</tr>
</tbody>
</table>

Record 1 of 3
**Product Sales - Frame 25**

This frame allows the end-user to schedule product sales to customers. When a product sale is entered, the specified volume is subtracted from the product schedule entries (FIFO), and the product schedule entries are marked as sold.

![Product Sales Frame]

**Help**

User help is activated from the main menu, "Help" menu item. Textual user help is available for each of the CROP modules.

**Help About**

This box is informational only.

![About Window]
Procedure - Summary

The CROP software provides an interface for the end-user to maintain refinery, crude oil, and product information in a relational database. The modules of the CROP software read, manipulate, update, insert, and report the data stored in the relational database. Each module accesses multiple tables, and creates or updates information that is used by the other modules. The LP uses basic crude and product information to create records that are used by the SD module. The SD module selects crudes for the Scheduling module. The Scheduling module is responsible for maintaining crude and product movements. The GMD module reads data created by the other modules and provides profit feedback to the user.

A consistent user-interface is maintained throughout the CROP software. Database procedures are triggered by entering rows into tables (ex: LP, Simulation, Purchase, Crude Run). The architecture of the application allows the end-user to view several modules at the same time.