Oil Industry Supply Chain Management Using English Business Rules Over SQL

Ted Kowalski, Shell Oil US
ted.l.kowalski@shell.com
Adrian Walker, Reengineering Llc.
ibl@snet.net

1. Introduction

When a customer region has a demand for a quantity of an oil product, it is in general possible to meet the demand using a number of equivalent products. Many factors influence the proportions of component products that are combined to make an optimal supply chain decision. The factors include the season of the year, the locations of available equivalent products, and the availability of suitable and timely transportation.

A competitive supply chain plan depends on knowledge of the above factors, on business policy knowledge, and on inventory facts in SQL databases. Both the knowledge and the facts can change rapidly. This makes it difficult to write conventional application programs and SQL queries that can produce supply chain plans to profitably meet demand.

We show how the knowledge needed to satisfy a target region demand can be written down in the form of business rules in open vocabulary, executable English. We use some new technology to directly execute the rules as though they were a program. The technology automatically generates and runs SQL queries to produce a suggested supply chain solution. Even in simple examples, the generated SQL queries are too complex for a programmer to write reliably. However, it is easy to change the business rules to specify a new policy, and the generated SQL then changes automatically. A feature of the technology is that a supply chain solution can be explained, at the business level, in hypertexted English.

The next section describes a supply chain example. Section 3 shows how the knowledge for finding a competitive supply chain plan can be written as business rules in executable English, and provides some small sample data tables. Section 4 shows how the rules are run over the sample data in a file to produce a supply chain solution, and to produce an English explanation of how the solution was calculated. In general, the approach in Section 4 would not scale up for large tables of data. In Section 5, we show how the rules can be used to automatically generate and run SQL queries that would be too complex for a programmer to write reliably. The resulting supply chain solution and explanation is the same as for the non-SQL approach in Section 4. We conclude that English business rules that can easily be changed, and that are used to generate and run SQL automatically, have the potential to increase the efficiency and competitiveness of supply chains in the oil industry.

2. A Supply Chain Example

We project that the target region NJ will need 1000 gallons of product 'y' in October, 2005. We then ask what alternative routes and modes-of-transportation (truck, train, boat, pipe) do we have to get that product to the region. Next we ask whether there's a refinery nearby that can produce the base product for finished product 'y'. With all of that, we finally say that we need a delivery plan that is optimized to deliver on time, make a profit, and beat the competition.

However, if there is not enough of product 'y', then, depending on the region and the customers, product 'x' or 'z' will do as well; they're just variations of ‘y’ using different additives. But they'll only do just as well in region NJ for the season including October 2005. This makes sales projections and marketing more complicated, but also gives us more competitive flexibility.

3. Business Rules in Executable, Open Vocabulary English, and Sample Data

This Section shows how the knowledge needed for the supply chain example can be written as business rules in executable English, and provides some small sample data tables. Section 4 will then show how the rules can be run to produce a supply chain plan, and to produce an explanation of how the solution was found.

Each of the business rules is in a form reminiscent of a classical syllogism. There are several premises, then a line, then a conclusion that can be drawn if each of the premises holds. Here is the first rule.

estimated demand some-id in some-region is for some-quantity gallons of some-finished-product in some-month of some-year
for estimated demand that-id some-fraction of the order will be some-product from some-refinery
*that-quantity * that-fraction = some-amount*
---
for demand that-id that-region for that-quantity that-finished-product we use that-amount that-product from that-refinery

The first premise looks to a data table for information about estimated demand in a region. The second premise looks to further rules that will determine the fractions of base and finished products used meet the demand. The third line calculates the amount of each component product that will be used . Then, the conclusion summarizes the planned allocation of quantities to meet the demand.

In the rule, *some-region*, *some-finished-product* and so on are place holders (or variables) that will be filled in with actual values when the rule is run. The remaining text, such as *we use* is open vocabulary English that ensures that we know the real world meaning of the rule. (Although the vocabulary is open, no dictionary or grammar maintenance is needed [2].)

The above rule depends on a second rule like this.

estimated demand some-id in some-region is for some-quantity gallons of some-finished-product in some-month of some-year
for demand that-id for that-finished-product refinery some-refinery can supply some-amount gallons of some-product
for demand that-id the refineries have altogether some-total gallons of acceptable base products
that-amount / that-total = some-long-fraction
that-long-fraction rounded to 2 places after the decimal point is some-fraction
---
for estimated demand that-id that-fraction of the order will be that-product from some-refinery

The rule specifies a policy for how much we will draw from each refinery that has an acceptable product. The policy says that we draw from each refinery according to its schedule of committed production. Of course, we could at this point try out other policies simply by changing the rules. (The SQL that the technology can generate would then change automatically to implement the new policy.)

The above rule depends on a further rule like this.

estimated demand some-id in some-region is for some-amount gallons of some-product in some-month of some-year
sum a-num : for demand that-id for that-product refinery some-name can supply some-num gallons of some-product1 = a-total
---
for demand that-id the refineries have altogether that-total gallons of acceptable base products

The next rules say how we qualify alternative products to meet the estimated demand:

estimated demand some-id in some-region is for some-quantity gallons of some-finished-product in some-month of some-year
in that-month an order for that-finished-product can consist in whole or part of some-base-product
in that-month the refinery some-name has committed to schedule some-amount gallons of that-base-product
we have some-method transportation from refinery that-name to region that-region
---
for demand that-id for that-finished-product refinery that-name can supply that-amount gallons of that-base-product

estimated demand some-id in some-region is for some-quantity gallons of some-finished-product in some-month of some-year
in that-month the refinery some-name has committed to schedule some-amount gallons of that-finished-product
we have some-method transportation from refinery that-name to region that-region

```
-------------------------------------------------------------------------------------------------------------------------------------------
for demand that-id for that-finished-product refinery that-name can supply that-amount gallons of that-finished-product
```

One more rule completes the executable English specification of the application:

```
for the purpose of this study, the current season is some-season
some-month is within the season that-season for production purposes
in that-season an order for some-finished-product can be filled with the alternative some-base-product
-------------------------------------------------------------------------------------------------------------------------------------------
in that-month an order for that-finished-product can consist in whole or part of that-base-product
```

At this point, we have specified the application using the above rules. However, we also need some sample data. For small amounts of test data, we can write tables like this one.

estimated demand this-id in this-region is for this-quantity gallons of this-finished-product in this-month of this-year

523	*NJ*	*1000*	*product-y*	*October*	*2005*

The table has an English heading, then a double underline, then a row of data. To complete the test data we need some more tables like this.

we have this-method transportation from refinery this-name to region this-region

truck	*Shell Canada One*	*NJ*
rail	*Shell Canada One*	*NJ*

in this-month the refinery this-name has committed to schedule this-amount gallons of this-product

October	*Shell Canada One*	*500*	*product-y*
October	*Shell Canada One*	*300*	*product-x*
October	*Shell Canada One*	*800*	*product-z*
October	*Shell Canada One*	*10000*	*product-w*

in this-season an order for this-product1 can be filled with the alternative this-product2

Fall	*product-y*	*product-x*
Fall	*product-y*	*product-z*

this-month is within the season this-season for production purposes

October	*Fall*
November	*Fall*
December	*Fall*

for the purpose of this study, the current season is this-season

Fall

We have now finished specifying an application together with its test data. The application is on the web, where it can be viewed and run using a browser [1]. In the next section, we describe what you would see if you did this.

4. Running the Business Rules Using a Small Data Sample

In the last section, we described a complete application together with a small amount of test data. In this section, we show what happens when the application is run using a browser.

Since we deal for the moment with a small amount of data, we do not need to use a SQL database. The next section will show that the same results can be obtained if we put the sample data into a DBMS, such as Oracle. The system then generates runs some complex SQL queries that would be difficult for a programmer to write accurately.

If we use a browser as described in [1] with the rules and sample data, the underlying Internet Business Logic system [2,3] presents us with a menu of questions we can ask. (We can also type in an English question, and the system will rank the menu of questions it knows how to answer accordingly.)

In this case, the top question is simply the conclusion of the first rule that we wrote, namely

for demand some-id some-region for some-quantity some-finished-product we use some-amount some-product from some-refinery

When we click on that sentence and then on the 'Ask' button, the system understands this as a request to find an answer table with the sentence as a heading. The result is an answer table like this:

for demand this-id this-region for this-quantity this-finished-product we use this-amount this-product from this-refinery

523	*NJ*	*1000*	*product-y*	*190.0*	*product-x*	*Shell Canada One*
523	*NJ*	*1000*	*product-y*	*310.0*	*product-y*	*Shell Canada One*
523	*NJ*	*1000*	*product-y*	*500.0*	*product-z*	*Shell Canada One*

Since we specified the application as rules in English, the system can explain, step by step in English, how it found any line of the answer table.
The explanation for the first line of the answer starts out like this.

estimated demand 523 in NJ is for 1000 gallons of product-y in October of 2005
for estimated demand 523 0.19 of the order will be product-x from Shell Canada One
*1000 * 0.19 = 190*
---
for demand 523 NJ for 1000 product-y we use 190 product-x from Shell Canada One

If we were to view the explanation on the web, we would see that it contains hypertext links, so that we can selectively drill down into the following details of how the answer was found.

estimated demand 523 in NJ is for 1000 gallons of product-y in October of 2005
for demand 523 for product-y refinery Shell Canada One can supply 300 gallons of product-x
for demand 523 the refineries have altogether 1600 gallons of acceptable base products
300 / 1600 = 0.1875
0.1875 rounded to 2 places after the decimal point is 0.19
---
for estimated demand 523 0.19 of the order will be product-x from Shell Canada One

estimated demand 523 in NJ is for 1000 gallons of product-y in October of 2005
in October an order for product-y can consist in whole or part of product-x
in October the refinery Shell Canada One has committed to schedule 300 gallons of product-x
we have truck transportation from refinery Shell Canada One to region NJ
---
for demand 523 for product-y refinery Shell Canada One can supply 300 gallons of product-x

estimated demand 523 in NJ is for 1000 gallons of product-y in October of 2005
sum eg-num : for demand 523 for product-y refinery eg-name can supply eg-num gallons of eg-product1 = 1600
---
for demand 523 the refineries have altogether 1600 gallons of acceptable base products

for the purpose of this study, the current season is Fall
October is within the season Fall for production purposes
in Fall an order for product-y can be filled with the alternative product-x

---
in October an order for product-y can consist in whole or part of product-x

for demand 523 for product-y refinery Shell Canada One can supply 300 gallons of product-x
for demand 523 for product-y refinery Shell Canada One can supply 500 gallons of product-y
for demand 523 for product-y refinery Shell Canada One can supply 800 gallons of product-z
---
sum eg-num : for demand 523 for product-y refinery eg-name can supply eg-num gallons of eg-product1 = 1600

There are some further explanation steps, such as

estimated demand 523 in NJ is for 1000 gallons of product-y in October of 2005
in October an order for product-y can consist in whole or part of product-x
in October the refinery Shell Canada One has committed to schedule 300 gallons of product-x
we have truck transportation from refinery Shell Canada One to region NJ
---
for demand 523 for product-y refinery Shell Canada One can supply 300 gallons of product-x

Finally, the explanation ends with

for the purpose of this study, the current season is Fall
October is within the season Fall for production purposes
in Fall an order for product-y can be filled with the alternative product-z
---
in October an order for product-y can consist in whole or part of product-z

So far, we have just used a small amount of test data, stored in a file together with the rules that define the application. For a real application, we would expect large amounts of data, and for this purpose we would need the efficiency and scalability of a DBMS such as Oracle. The next Section describes how to do this.

5. Using the Rules to Automatically Generate and Run Complex SQL Queries

Section 3 described some rules that specify a way of combining several products to meet an estimated demand, and Section 4 showed how the rules can be run over a small sample of data in a file. However, we said that the approach would not scale efficiently to large tables of data. In this Section we show how to use the same rules in the same Internet Business Logic system, with the data now residing in a DBMS. To do this, the system automatically generates and runs SQL queries. The DBMS containing the data can be anywhere on an intranet or on the web, provided the appropriate network and query permissions are granted.

To run the application using SQL, the rules remain the same. However, the tables are replaced by special rules that say where on a network to find the data. For example, the table

we have this-method transportation from refinery this-name to region this-region
==========

truck	*Shell Canada One*	*NJ*
rail	*Shell Canada One*	*NJ*

is replaced by a special linking rule like this

url:12.34.56.789 dbms:9i dbname:ibldb tablename:T1 port:1521 id:anonymous password:oracle
---
we have this-method transportation from refinery this-name to region this-region

The premise of the rule says that we can find the data at the network address 12.345.6.789 in an Oracle 9i DBMS. The data is a table named T1. The network port to use in number 1521, with the id 'anonymous' and the password 'oracle'.

The other tables are replaced by similar linking rules.

The rules in section 3 that specify the application turn out to have a surprising amount of logical power when automatically translated into SQL. When we ask the question

for demand some-id some-region for some-quantity some-finished-product we use some-amount some-product from some-refinery

the Internet Business Logic system generates and runs four SQL queries.

Here is one of the generated queries.

```
select distinct x6,T2.PRODUCT,T1.NAME,T2.AMOUNT,x5 from
  T6 tt1,T6 tt2,T5,T4,T3,T2,T1,T6,
  (select x3 x6,T6.FINISHED_PRODUCT x7,T6.ID x8,tt1.ID x9,tt2.ID x10,sum(x4) x5 from
    T6,T6 tt1,T6 tt2,
    ((select T6.ID x3,T3.PRODUCT1,T1.NAME,T2.AMOUNT x4,T2.PRODUCT from
      T1,T2,T3,T4,T5,T6,T6 tt1,T6 tt2 where
        T1.NAME=T2.NAME and T1.REGION=T6.REGION and T2.MONTH1=T4.MONTH1 and
        T2.MONTH1=T6.MONTH1 and  T2.PRODUCT=T3.PRODUCT2 and T4.MONTH1=T6.MONTH1 and
        T3.PRODUCT1=T6.FINISHED_PRODUCT and T3.SEASON=T4.SEASON and T3.SEASON=T5.SEASON and
        T4.SEASON=T5.SEASON and T6.ID=tt1.ID and  T6.ID=tt2.ID and tt1.ID=tt2.ID)
     union
     (select T6.ID x3,T2.PRODUCT,T1.NAME,T2.AMOUNT x4,T2.PRODUCT from
      T1,T2,T3,T4,T5,T6,T6 tt1,T6 tt2 where
        T1.NAME=T2.NAME and T1.REGION=T6.REGION and T2.MONTH1=T6.MONTH1 and
        T2.PRODUCT=T6.FINISHED_PRODUCT and T6.ID=tt1.ID and T6.ID=tt2.ID and tt1.ID=tt2.ID)
    )  group by T6.FINISHED_PRODUCT,T6.ID,tt1.ID,tt2.ID,x3) where
     T6.ID=tt2.ID and tt1.ID=T6.ID and T6.FINISHED_PRODUCT=x7 and T6.ID=x8 and tt1.ID=x8 and
     tt2.ID=x8 and T1.NAME=T2.NAME and T1.REGION=tt2.REGION and T2.MONTH1=T4.MONTH1 and
     T2.MONTH1=tt2.MONTH1 and T2.PRODUCT=T3.PRODUCT2 and
     T3.PRODUCT1=tt1.FINISHED_PRODUCT and T3.PRODUCT1=tt2.FINISHED_PRODUCT and
     T3.SEASON=T4.SEASON and T3.SEASON=T5.SEASON and T4.MONTH1=tt2.MONTH1 and
     T4.SEASON=T5.SEASON and T6.ID=x6 and tt1.FINISHED_PRODUCT=tt2.FINISHED_PRODUCT and
     tt1.ID=tt2.ID and tt1.ID=x6 and tt2.ID=x6
       order by x6,T2.PRODUCT,T1.NAME,T2.AMOUNT,x5;
```

It would be difficult to write the above SQL query by hand, or to manually reconcile with the business knowledge specified in the rules. However, as in case when SQL is not used, we can get a business-level explanation of each line the result.

6. Conclusions

Combining a number of oil products in a supply chain to meet an estimated demand depends a number of factors. The knowledge and data required to do this can change at a faster pace than can be tracked by conventional application programming and manual writing of SQL queries.

We have described an example of using a system that supports the writing and running of business rules in open vocabulary, executable English. In the example, an estimated demand for a particular finished product is met with a number of equivalent products according to a policy specified in business rules. It is easy to change the policy rules at the business level, using a browser. Hypertexted English explanations of the results are available on demand. The example and the system are live on the web, and the author- and user interface is simply a browser.
The system can operate directly on small amounts of test data in a file. For larger amounts of data, the system can automatically generate and run SQL queries over a network. For the example given, the generated is SQL is too complex to be written by hand, and it would be difficult to manually reconcile it to knowledge at the business level without the provided end user level explanations. However, a change in policy at the business level automatically changes the generated SQL, and English explanations can be used to check the results.

Business rules in English that can easily be changed, and that are used to generate and run SQL automatically, have the potential to increase the efficiency and competitiveness of supply chains in the oil industry.

7. References

[1] The example described in this paper can be viewed, run and changed as follows:

1. Use a browser to go to the page http://www.reengineeringllc.com
2. click on Internet Business Logic
3. click the GO button
4. select the example *Oil-IndustrySupplyChain1* from the list in the middle of the page
5. at the top of the page, click on "Choose an Agent and Go to its Question Menu"
6. click elsewhere on the page
7. you should now see a Question Menu
8. click on the first sentence
9. you should now see a new window with an "Ask" button
10. click the Ask button
11. you should now see an Answer Table
12. click on "Go To the Question Menu" hold down the mouse button, select "Get an Explanation of the Selected Line" and release the button
13. you should now see a step-by-step explanation of how the system used the rules and facts in the example to get the answer
14. at the top of the page, choose "Go to View or Change the Agent" you will see the rules and facts that you have just been running
15. please use the Help button on each page to see how to navigate further
16. the tutorials show how to write and run your own examples.

[2] A one slide overview of the Internet Business Logic system is at http://www.reengineeringllc.com/internet_business_logic_in_a_nutshell.pdf

[3] For more detail on the architecture of the Internet Business Logic system, please see slide 14 of http://www.reengineeringllc.com/Internet_Business_Logic_Presentation.pdf

20050526