An SQL-based cost-effective inventory optimization solution

K. Katircioglu
T. M. Brown
M. Asghar

Successful implementation of an inventory optimization solution requires significant effort and can pose certain risks to companies implementing such solutions. Depending on the complexity of the requirements, the solution may also involve a substantial IT investment. In this paper, we present a cost-effective solution for inventory optimization that can be useful for small and medium-sized businesses with limited IT budgets. This solution can be implemented on any application platform that is capable of processing basic SQL™ (Structured Query Language) commands. The solution eliminates the need to purchase additional software and has a framework in which sales data in an Enterprise Resource Planning (ERP) system are accessed, demand statistics based on this data are generated along with other key parameters, and optimal inventory policies, such as those involving safety stocks and lot sizes, are calculated and reported.

Introduction

One of the most tangible ways in which companies can benefit from IT investments is through implementing solutions that are designed to provide inventory reduction opportunities. Companies can reduce their inventory through more accurate and faster receipt of demand information, more reliable and shorter supply lead times, and better inventory policies.

Faster receipt of demand information and shorter lead times can be achieved by changing planning processes, modifying supply chain infrastructure, and adopting practices that enable rapid flow of information on both the demand and supply sides. Supply chain management (SCM) solutions are designed to support these activities. (The term *lead time* refers to the amount of time between the placing of an order and the receipt of the goods ordered.) Demand-planning solutions can also help increase the accuracy of demand forecasts.

Improved reliability of supply arrival times also has an impact on inventory levels. Because companies must keep a safety stock to protect against variations in either supply quantity or supply arrival times, more reliable supply arrival times can reduce these safety stocks. (The term *safety stock* refers to the extra units of inventory carried by a company as protection against possible stock outages.) Clearly, shorter supply lead times also reduce the need for safety stock.

On the other hand, no matter how well the planning processes are executed or how advanced they may be, some uncertainty almost always remains in a typical supply chain on both the demand side and the supply side. Companies continue to have a need for inventory policies that are appropriate for their supply chain environment and protect them from the serious consequences of these uncertainties. For instance, a good inventory policy takes into account demand variation, supply lead times, and the service objective of a company and calculates the size of safety stocks that can provide the targeted service level with the minimum possible inventory.

Inventory management is a well-established field of research in both the academic and the industrial worlds, and numerous publications and popular books are available [1–8]. *Analysis of Inventory Systems* is an early classic and a valuable source [9].

Numerous academic papers have made significant contributions to the field. While it is not possible to include all without providing an extensive survey, we can mention [10–16] as examples of classic publications in the field, and [17–19] as examples of later publications that have made important contributions.

Optimized inventory policies are the most important output provided by inventory optimization software. Such policies are often characterized by a few

0018-8646/07/$5.00 © 2007 IBM

parameters such as safety stock and lot size. Optimal inventory policies can reduce inventory investment and improve customer service significantly. Hence, most of the time, a very attractive business case exists for purchasing and implementing such inventory software. Of course, companies that do not have the execution processes in place for proper implementation of these policies cannot benefit from them. Along with other basics, execution typically requires the visibility of different kinds of inventories (such as materials on hand, on order, committed, or in transit) across the supply chain, as well as the visibility of demand information such as forecasts and sales orders. Most Enterprise Resource Planning (ERP) systems or SCM systems are capable of providing this visibility, although fragmented systems—for example, in the form of IT systems with little integration or with different applications—can lead to data-integrity problems, such as data with missing fields or with different definitions.

As is the case for any other software solution, some concerns may exist with respect to using standalone software to optimize inventory. (Note that when we use the term *standalone software*, we refer to software that is designed to be self-sufficient in terms of performing a process or a part of the process, and that usually includes such basic elements as a database, capability to store and manipulate inputs and outputs, an engine that performs the calculations, and a graphical user interface.) If many elements of an inventory management process are missing, a standalone software package may be the right choice for a company, because such a package can help establish and facilitate the entire process. However, if many of the elements already exist, and a company has to focus on optimizing safety stocks or lot sizes, a quick and inexpensive solution like ours can be a viable choice. Our solution is designed as a part of the ERP system and hence does not require complex integration. Typical concerns associated with standalone software use are outlined in the following paragraphs.

Software proliferation: In order to gain full benefit of a well-planned and well-executed supply chain process, companies often find themselves in a situation in which they must buy several layers of applications. Typically, their ERP system may be considered to be the bottom layer. Systems or applications may be integrated into ERP. These include applications for distribution requirements planning (DRP), materials requirements planning (MRP), demand planning, SCM, order fulfillment, procurement planning, and inventory-optimization systems. These applications, which may have complex integrations with one another, present a serious IT challenge to companies that are concerned about cost-effective management and maintenance of their IT portfolios. If major business process changes evolve and the applications are therefore required to support these changes, they may be inadequate or may require substantial modifications to deliver the required functionality. If the process changes are so extensive that they affect a number of applications and the way they must integrate, it may be very expensive to perform the necessary changes in these applications. Realizing that these costs can be very high, many companies understandably resist additions to their systems portfolio because they do not know what changes will be required in the future.

Cost of implementation: Standalone solutions must address issues such as Internet access infrastructure, security, and system integration. Such concerns lead to an unavoidable infrastructure cost that has to be incurred in order to realize the benefit of the solution, which facilitates inventory reduction and/or customer service improvements. To the extent that a solution can take advantage of existing infrastructure in addressing these issues, it will be more attractive from an implementation and maintenance cost point of view.

Systems integration: Since inventory optimization has extensive data requirements, integration into existing ERP and SCM systems can be time-consuming and costly. However, robust and flexible data-extraction templates can significantly reduce system integration costs. Note that when major version updates occur, version compatibility can become problematic while maintaining this integration. Complex forms of integration are clearly undesirable because business changes may also require modifications in the way applications must integrate.

Purchase price, maintenance and user fees: Standalone software typically requires an up-front purchase cost, annual fees based on the number of users, and maintenance contracts. Version upgrades add to these costs, as does training and/or hiring new staff who can use the software effectively.

Cost-effective inventory optimization solution

Most of the up-front effort, and therefore cost, incurred with inventory optimization software comes from data and systems integration. Although the algorithmic part of a solution that performs the inventory optimization provides value, this is relatively inexpensive because many algorithms with a range of features are readily available from various sources such as textbooks, handbooks, and academic publications. Some very specialized software companies or companies that have strong research departments can provide proprietary techniques that are not publicly available.

The approach used by most standalone inventory optimization software packages may be described as follows. First, the package obtains an extract of data

from an ERP system (typically via a flat file, containing records with no structured relationships, or a Microsoft Excel** file). The data is imported to a software package and converted to a form that an optimization engine can use. The calculations are carried out using a language such as C or C++. The output is placed in a data repository that can be viewed by the user and sent back to the ERP system or an Advanced Planning and Scheduling (APS) system.

In many cases that are not complex, relatively simple algorithms may provide satisfactory solutions. In fact, the solution-finding process of some common inventory problems can be simplified to the extent that these solutions can be calculated using simple mathematical functions available in Structured Query Language (SQL).

SQL is a language that enables access to a database, performs queries with flexible conditions, and facilitates basic data processing operations using commands such as *retrieve*, *insert*, *delete*, and *update*. Although many versions of SQL exist, basic commands in compliance with American National Standards Institute (ANSI) standards are supported by all providers. SQL is commonly available in existing database software offered by various vendors and widely used by organizations for managing large sets of data. An accessible introduction to SQL can be found in [20].

Although the wide availability of SQL is an advantage, it does have limitations. For tasks such as inventory optimization, SQL is not the appropriate tool, since it is not designed for programming and executing advanced algorithms during the runtime of an application. However, we can simplify some common inventory optimization problems in order to perform the optimal safety stock and lot size (that is, economic order quantity) calculations using basic mathematical functions available in SQL. (The term *economic order quantity* refers to a quantity that minimizes the fixed inventory replenishment costs and inventory carrying costs.) Coupled with readily available data-extraction templates in many database software packages, our approach makes it possible to create simple inventory optimization capabilities for databases.

To take advantage of SQL, we attempted to make our approach work seamlessly for systems with databases that have strong capabilities for extracting and manipulating data from transaction systems. **Figure 1** schematically illustrates how our approach works. Using available data extractors, we extract, transform, and load (ETL) essential data such as sales transactions and lead times into multidimensional summary data tables called InfoCubes in the SAP** literature. [SAP, which stands for systems, applications, and products in data processing, is a business application and an ERP solution software provider.] After the ETL step, we can calculate optimal inventory policies using query commands in runtime.

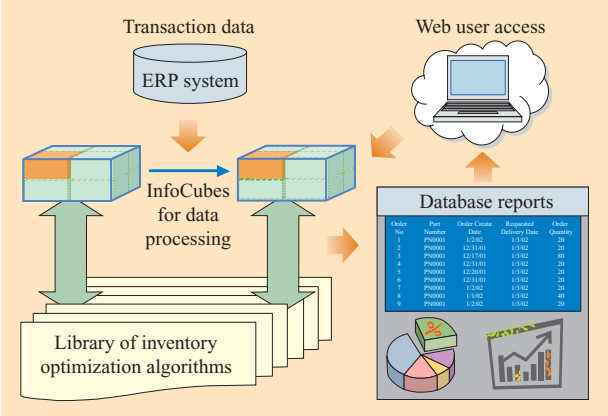

Figure 1

Simple approach for inventory optimization using SQL. Arrows show the directions of information flows. Green double arrows represent the closely integrated InfoCubes and algorithms. All calculations are performed using query commands, which are closely tied to the data in the InfoCubes.

Database management systems (DBMSs) are designed to maintain data and answer queries that are typically executed through SQL. Queries can be real-time operations, and they allow real-time interaction with the users. However, this interaction is mainly for the purpose of reporting basic statistics about the data, such as total sales in a period or total inventory for a product group or a geographical region. Complex operations, such as algorithms used to optimize inventory policies, normally cannot be performed by queries. Although programming languages such as C, C++, and Java** are certainly capable of performing such operations, they cannot be parts of query commands that must process significant amounts of input data in runtime. Java-based programs may be able to perform calculations with acceptable speed. However, speed may be lost in open database connectivity (ODBC) and Java database connectivity (JDBC) layers that are necessary for a Java inplementation. For runtime efficiency, these calculations must be performed outside the query commands in batch mode. Obtaining results this way requires a few steps, as we described earlier, and the real-time nature of the interaction is lost. In addition, file downloads from and uploads to databases can be time-consuming. The ability to perform optimization in the queries provides users with interaction capabilities and enables them to do real-time "what-if" analyses in addition to avoiding file downloads and uploads. Most significantly, the optimization speed can be acceptable.

In a solution that we designed on a common ERP platform, we were able to run queries in order to calculate

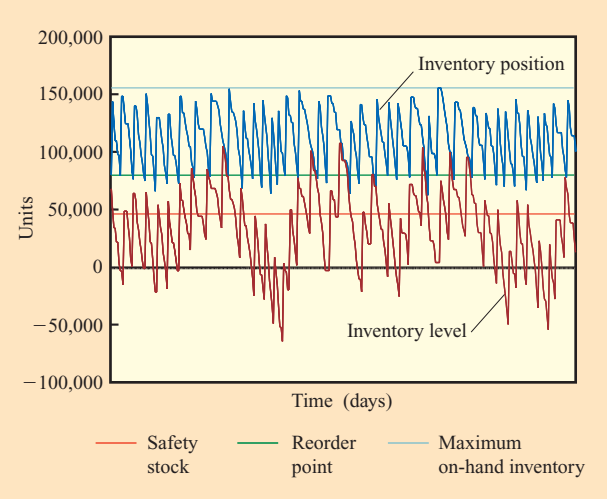

Figure 2

Simple inventory policy in action. When the inventory position decreases to the reorder point, a replenishment order is placed. The replenishment order has a calculated lot size, and until the order arrives, inventory shortages are prevented through the use of safety stocks. The inventory that is on hand can never exceed the maximum on-hand inventory. (Inventory level: on-hand inventory minus backlogs. Backlogs: demand that is waiting to be satisfied because of lack of inventory.)

optimal inventory policies and more than 30 related parameters in 25 seconds for ten products with a total of 10,000 customer order transactions constituting demand data. We used an IBM iSeries* server with an OS/400* operating system running on four processors. The data was stored in a DB2* database. Our solution design had several advantages, as listed in the following paragraphs.

First, we note that our SQL-based approach was simple in that SQL commands are communicated to the database layer without implementing the ODBC and JDBC connections and drivers required by a Java-based solution. Data is efficiently retrieved from an already established data model. This architecture also efficiently and flexibly enables characteristics-based reporting on the data.

Second, we note that since the ERP system we used also had a flexible query-reporting capability based on a data warehouse (i.e., a primary repository of an organization's historical data), no additional vendor or software products were needed to gather data from the back-end databases and communicate the results to the web applications. More generally, our approach allows us to calculate additional parameters by adding their formulas to the query commands without having to modify underlying programs such as Java code. By using the web server that is native to the available ERP system, we needed no additional application development in order to display the results of queries in the form of charts and tables. A Java application typically requires the creation of an HMTL document in order to communicate the results over the Web.

We next present some classes of inventory problems and show how their solutions can be simplified and implemented in SQL.

Some common inventory problems and their solutions

Although many inventory optimization problems may require relatively complex algorithms, some classes of inventory problems can be simplified using approximations. We focus on simple inventory policies in which a safety stock and lot size characterize the inventory policy. **Figure 2** shows how a simple inventory policy works. As discussed previously, safety stock is the stock that is held in order to protect against uncertainties in the process, such as uncertainty in demand and uncertainty in supply lead time. When the inventory position (defined as the inventory on hand plus inventory on order minus the back orders) drops to the reorder point, inventory is replenished at an amount equal to the lot size. Most often, this replenishment is not instant (i.e., orders arrive after a lead time).

Next, we study different types of common inventory problems that have simple solutions. Before we proceed, we need to mention that the calculations of optimal inventory policies that we present here are based on the following assumptions.

Our model assumes that inventory is held at a single location. This model can also be used to approximate multiple-location models. A single-location model is one in which inventory is held at a single location. Each SKU has independent and identically distributed demand in each time period. (The term *SKU* stands for stock-keeping unit; it is an identifier that helps merchants track products and services.) We assume a time period that corresponds to a day. Customer orders arrive according to a Poisson distribution. The quantity requested in an order is random and follows a normal distribution. The quantities of all orders are independent, identically distributed random variables. Demand that occurs during the replenishment lead time has (or is approximated by) a normal distribution. Inventory is replenished through manufacturing or a supplier with a positive lead time. In the case in which the inventory is reviewed periodically, the service targets are set for the worst period in the replenishment cycle. Therefore, the actual inventory availability levels might be higher than the target. This is a conservative approach; if some of the assumptions made in the model do not hold, this approach could compensate for a potential shortfall in

actual service levels. Calculating optimal policies for average service criteria requires search algorithms. Although these algorithms would be simple to code in a language such as C, it is not possible to code loops in SQL. Therefore, we restrict service targets to the conservative scenario. In the case in which inventory is reviewed continuously, service targets are set to control the average performance during a cycle.

The objective is one of the following: 1) minimize expected inventory costs; 2) maximize expected profit; 3) achieve target service with minimum expected inventory; or 4) calculate safety stock for a fixed "days-of-supply" policy. *Days of supply* is a measure of inventory level, calculated by dividing inventory in units by the average demand per day.

The service is defined by one of the following: 1) the probability of no stock-out (i.e., the probability of inventory being available to satisfy demand); 2) the probability of on-time shipment to meet the customer request date; 3) the probability of on-time shipment to commit (i.e., the probability that shipment was made no later than the date that was committed to the customer); and 4) the fill rate (which relates to the fraction of demand that can be met from inventory with no delay).

The lot-sizing policy is defined by one of the following: 1) fixed lot size (provided by the user); 2) variable lot size (the quantity that brings inventory position to a specified level); or 3) economic order quantity (the optimal order quantity that minimizes the sum of fixed order cost and inventory carrying cost). The inventory-replenishment process is defined by continuous review and replenishment or periodic review and replenishment.

Notation

In this section, we introduce the parameters that we need in order to perform the calculations of optimal lot size and optimal safety stock that make up an optimal inventory policy. As we introduce these parameters, we indicate whether they are considered as input, intermediate calculations, or output. Some parameters can be provided as inputs or can be calculated on the basis of their historical observations. We label such parameters as *input or intermediate calculations*. In the lists, we follow an order of precedence; that is, inputs come first, followed by intermediate calculations and outputs.

Inputs

PNS: Inventory availability target in percentages (probability of no stock-out, i.e., the event of running out of inventory).

FRT: Fill rate target in percentages, i.e., the fraction of demand that can be met from inventory without any delay.

OTDR: On-time delivery to customer request target, in percentages.

OTDC: On-time delivery to commit target, in percentages.

OCOST: Fixed order cost (per order).

SCOST: Shortage cost (per unit per period).

HCOST: Inventory holding cost (per unit per period).

COST: Cost of a unit (cost of manufacturing or purchase).

PRICE: Price of a unit.

SVALUE: Salvage value of a unit when season is over (for seasonal items). This value is the price offered when trying to dispose of the unsold inventory, and this price is almost always a highly discounted one.

PDECLINE: Price decline of a unit (during a season or period).

QFIX: Fixed lot size given by the user.

QUNIT: Unit lot size. (The output parameter *LOT*, which represents lot size, must be an integer multiple of this quantity.)

QMIN: Minimum lot size.

QMAX: Maximum lot size.

T: Number of days in the data time window. (Note: The length of the time window must be determined prior to calculating *T*.)

N: Number of orders during *T* days (i.e., during the entire data time window).

NDAYS: Number of working days in a year.

I: Order number in the data window ($I = 1, 2, \cdots, N$).

J: Day number in the data window ($J = 1, 2, \cdots, T$).

DOQ[*J*]: Daily order quantity [total order quantity in day *J* ($J = 1, 2, \cdots, T$)].

ORQ[*I*]: Order quantity in order *I* ($I = 1, 2, \cdots, N$).

TSERVICE: Target service level (percentage).

MLT: Manufacturing (or supply) cycle time or period length (in days) when inventory is reviewed and replenished periodically. This parameter is also called "manufacturing (or supply) frequency" and is the number of days during which supplies are regularly planned.

PLT: Manufacturing (supply) lead time (in days). This is the actual physical manufacturing time (or supplier's turnaround time).

OPT: Order processing time (in days). This is the number of days required to process the inventory replenishment orders from the stocking locations.

TLT: Transportation time (in days). This is the number of days required to transport orders from the manufacturing location to the stocking location.

IP: Inventory position. This is a key metric for the execution of any inventory policy. It must be monitored

during execution and not calculated in optimization. This requires monitoring inventory on hand, inventory on order, and amount of demand back-ordered. The proper implementation of *ROP* (reorder point) requires keeping track of the inventory position, because replenishment orders should be triggered only when the inventory position (not the inventory on hand) drops to *ROP*. *IP* is given by $IP = inventory\ on\ hand + inventory\ on\ order - back\ orders$.

Inputs or intermediary calculations

SPLT: Standard deviation of manufacturing (supply) lead time *PLT*.

SOPT: Standard deviation of order processing time *OPT*.

STLT: Standard deviation of transportation time *TLT*.

CLTR: Customer order lead time (requested by customer).

CLTC: Customer order lead time (committed to customer).

SCLTR: Standard deviation of customer order lead time *CLTR*.

Intermediary calculations

RLT: Replenishment lead time (in days).

SRLT: Standard deviation of replenishment lead time *RLT*.

ADLT: Average demand during replenishment lead time.

SDLT: Standard deviation of demand during replenishment lead time.

OAR: Order arrival rate (average number of orders per day).

TOQ: Sum of all order quantities.

TOQS: Sum of the squares of all order quantities.

TOD: Total of all daily demand quantities.

TODS: Total of the squares of all daily demand quantities.

AOQ: Average order quantity (i.e., average quantity demanded in a single customer order).

SOQ: Standard deviation of order quantity.

AOD: Average demand during a day.

SOD: Standard deviation of the amount of quantity ordered during a day.

PDURATION: Duration (in days) of a season (or a period) during which expected profit is to be maximized or expected cost is to be minimized.

k: Safety factor used for safety stock calculation.

PVALUE: z value for the standard normal distribution. This is used for the calculation of the safety factor *k*.

Outputs

SS: Safety stock.

ROP: Reorder point.

LOT: Lot size.

MAX: Maximum inventory level.

Throughout the calculations below, we use sqrt(X) to denote the square root of X, max(X, Y) to denote the maximum of X and Y, min(X, Y) to denote the minimum of X and Y, and ln(X) to denote the natural logarithm of X.

Calculation of optimal inventory policies using SQL functions

All of the calculations presented in the following are designed in such a way that they can be coded in SQL. Since, as discussed, query languages are not able to use loops and they have limited mathematical functions available, we provide approximate solutions for problems that normally require looping algorithms to solve.

Calculation of key problem parameters

First, we introduce different cases that apply to practical inventory problems. Then, in each case, we calculate three important parameters: *PVALUE, RLT*, and *SRLT*. Very briefly, *PVALUE* represents or summarizes the objective function of the optimization problem. The inverse of the cumulative distribution of demand during the total replenishment lead time at the *PVALUE* gives the optimal amount of safety stock. *RLT* and *SRLT* are respectively the mean and standard deviation of the demand during the total replenishment lead time. We refer to these three parameters as the *key problem parameters*.

Case 1. Minimize expected inventory costs

In Case 1, the objective is to minimize the expected inventory holding and backlogging costs. We assume that the demand not met immediately in a period is backlogged to the next period. Backlogging has costs associated with expediting orders, paying a penalty to the customer, buying inventory at a high price, or loss of customer goodwill. We use a "myopic solution" to this cost problem. This solution tries to minimize the expected inventory-carrying and backlogging costs one period at a time (hence our use of the term *myopic*). In general this solution is not optimal, and the optimal policies can be complex. However, the simplicity of the method makes it practical. Reference [13] provides conditions under which myopic policies are optimal. When significant obsolescence costs exist that are associated with unsold inventory, myopic policies may be unwise to use and caution is warranted. One may also study [21] for such cases.

In the following, we show the calculation of three key intermediary parameters, $PVALUE$, RLT, and $SRLT$:

A. If lot size is flexible and inventory is reviewed periodically, then

$$PVALUE = \frac{SCOST}{SCOST + HCOST}.$$

B. If a lot size LOT exists, then

$$PVALUE = \frac{LOT \times HCOST}{AOD \times NDAYS \times SCOST}.$$

A. If inventory is reviewed and replenished continuously (i.e., manufacturing or purchasing can be performed at any time), then

$$RLT = OPT + TLT + PLT.$$

B. If inventory review and supply replenishment are performed periodically with review period length MLT, then

$$RLT = OPT + TLT + PLT + MLT,$$

$$SRLT = \text{sqrt}\,(SOPT^2 + STLT^2 + SPLT^2).$$

Case 2. Maximize expected profit

In Case 2, the objective is to maximize the expected profit (revenue minus purchase cost minus inventory cost). This maximization is typically used in cases in which a large-quantity purchase (or manufacturing build) exists prior to a season, or a period such as a month or a quarter. The objective is to purchase or build the appropriate quantity in order to maximize the expected profit in that season or period. Since the supply quantity must be decided before observing the demand, a risk is associated with the supply quantity. Excess supply is sold at a loss at the end of the season or period. Supply shortage causes revenue and profit shortfall. Relevant parameters include $COST$ (purchase price or manufacturing cost per unit), $PRICE$ (price of the unit in the market), and $SVALUE$ (value per unit for any unsold units left at the end of the season or other period when salvaged).

When the problem has multiple periods, the salvage value per unit must be calculated as $SVALUE = PRICE - PDECLINE$. Here, $PDECLINE$ is the loss of value of a unit (or price decline) during the season or other period. For items that have a very long life, where there is no or minimal price decline, this model is not appropriate. Because a perpetual demand exists for such items, the focus is typically on cost minimization or service target achievement instead of profit maximization.

The $PVALUE$ that maximizes the expected profit during the season or other period is given by the following:

$$PVALUE = (PRICE - COST)/(PRICE - SVALUE),$$

$$RLT = PDURATION,$$

$$SRLT = SPDURATION.$$

Case 3. Achieve a target probability of no stock-out

In Case 3, the probability of no stock-out is defined by PNS (see notations above). $PVALUE$, RLT, and $SRLT$ are calculated as follows. First, $PVALUE = PNS$. Note that if inventory is reviewed and replenished continuously (i.e., manufacturing can be done at any time, or purchase can be done at any time), then $RLT = OPT + TLT + PLT$. If inventory review and supply replenishment are done periodically with review period length MLT, then

$$RLT = OPT + TLT + PLT + MLT,$$

$$SRLT = \text{sqrt}\,(SOPT^2 + STLT^2 + SPLT^2).$$

Case 4. Achieve a target probability of on-time shipment to customer request date

In Case 4, on-time delivery in response to a customer request is defined by $OTDR$ (see notations above). $PVALUE$, RLT, and $SRLT$ are calculated as follows. First, $PVALUE = OTDR$. If inventory is reviewed and replenished continuously (i.e., manufacturing can be done at any time, or purchase can be done at any time), then $RLT = OPT + TLT + PLT - CLTR$. If inventory review and supply replenishment are done periodically with review period length MLT, then

$$RLT = OPT + TLT + PLT - CLTR + MLT,$$

$$SRLT = \text{sqrt}\,(SOPT^2 + STLT^2 + SPLT^2 + SCLTR^2).$$

Case 5. Achieve target probability of on-time shipment to commit date

In Case 5, on-time delivery in response to a customer request is defined by $OTDC$ (see notations above). $PVALUE$, RLT, and $SRLT$ are calculated as follows. First, $PVALUE = OTDC$. If inventory is reviewed and replenished continuously (i.e., manufacturing can be done at any time, or purchase can be done at any time), then $RLT = OPT + TLT + PLT - CLTC$. If inventory review and supply replenishment are done periodically with review period length MLT, then

$$RLT = OPT + TLT + PLT - CLTC + MLT,$$

$$SRLT = \text{sqrt}\,(SOPT^2 + STLT^2 + SPLT^2).$$

Case 6. Achieve a target fill rate

In Case 6, fill rate is defined by FRT (see notations above). $PVALUE$, RLT, and $SRLT$ are calculated as follows. First, $PVALUE = FRT$. If inventory is reviewed and replenished continuously (i.e., manufacturing can

be done at any time, or purchase can be done at any time), then $RLT = OPT + TLT + PLT$. If inventory review and supply replenishment are done periodically with review period length MLT, then

$$RLT = OPT + TLT + PLT + MLT,$$

$$SRLT = \text{sqrt}\,(SOPT^2 + STLT^2 + SPLT^2).$$

Explanation of steps in calculations

We now give a step-by-step description of the method for calculating the optimal inventory policies by using the key problem parameters that we have calculated above.

Step 1: Calculate daily demand statistics

$TOQ = ORQ[1] + ORQ[2] + \cdots + ORQ[N]$
(sum of quantities of all orders in the data time window).

$AOQ = TOQ/N$ (average order quantity).

$TOQS = ORQ[1]^2 + ORQ[2]^2 + \cdots + ORQ[N]^2$
(sum of squares of all orders in the data time window).

$SOQ = \text{sqrt}\,[(TOQS - N \times AOQ^2)/(N-1)]$
(standard deviation of order quantity).

$OAR = N/T$ (average number of orders per day).

$AOD = OAR \times AOQ$ (average demand during a day).

$SOD = \text{sqrt}\,(AOD^2 + OAR \times SOQ^2)$
(standard deviation of demand during a day).

Step 2: Calculate lead-time statistics

Up to this point, in the above calculations we have assumed that total replenishment time RLT is a known and fixed quantity. If the total replenishment time is not fixed, and every time it can be a different number, one must keep track of this lead time for each item-location combination through time. Then, by using this data one can estimate the mean and standard deviation of this lead time.

We assume that all lead times (i.e., OPT, TLT, PLT, and $CLTR$) are random, except for the manufacturing cycle time MLT and the customer order lead-time commit $CLTC$.

In the following, we demonstrate how to calculate the standard deviation of the transportation lead time TLT if one has a sample of transportation lead times. These calculations can be used for other lead times for which a sample of observations is available. First we introduce the following notation:

TLT: Mean (or average) transportation lead time (as previously defined).

TLTS: Sum of the squares of transportation lead-time observations in the sample.

STLT: Standard deviation of transportation lead time (this must be calculated from sample data of transportation lead times for that item, from the plant to the stocking location for the item).

Consider a case with K transportation lead-time observations in our sample given by $L[1], L[2], \cdots, L[K]$. We then calculate TLT as follows:

$TLT = (L[1] + L[2] + \cdots + L[K])/K$
(average of all transportation times in the sample)

To calculate the standard deviation, we need the sum of the squares of transportation lead times, which we denote it as $TLTS$. In other words,

$TLTS = L[1]^2 + L[2]^2 + \cdots + L[K]^2$
(sum of squares of all transportation times in the sample).

We then calculate the standard deviation as follows:

$STLT = \text{sqrt}[(TLTS = K \times TLT^2)/(K-1)]$
(standard deviation of all transportation times in the sample).

Step 3: Calculate key problem parameters

For various different types of problems, earlier in this paper we have provided ways to calculate the key intermediary parameters $PVALUE$, RLT, and $SRLT$. These are used to calculate the optimal inventory policy parameters.

Step 4: Calculate mean and standard deviation of demand during lead time

Mean demand during the replenishment lead time is $ADLT = AOD \times RLT$. If the lead time is random with standard deviation $SRLT$, standard deviation of demand during lead time is $SDLT = \text{sqrt}\,(SRLT^2 \times AOD^2 + SOD^2 \times RLT)$. If lead time is fixed, standard deviation of demand during lead time is $SDLT = SOD \times \text{sqrt}\,(RLT)$.

Step 5: Calculate fixed lot size

All formulas provided so far are appropriate for one-for-one replenishment where no restriction exists on the lot size or the frequency of orders. When the lot size is fixed or a minimum or maximum restriction exists on it, because of replenishment process limitations, supply requirements, or cost considerations, these restrictions must be taken into account.

The user provides a fixed lot size as an input, and this size is used directly in the calculations. In this case, the inventory policy is still to bring "inventory position" to ROP level whenever it falls below ROP. However, because of the restriction on the lot size, the order quantity must reflect these restrictions. We have five different types of lot size restrictions, discussed in the following sections.

Case 1: Fixed lot size

In this case, there is a predetermined lot size $QFIX$. Whenever the inventory position drops to ROP,

this quantity is ordered. Here, $LOT = QFIX$. If no predetermined lot size exists and economic order quantity cannot be calculated because of the lack of fixed order cost data (or if economic order quantity is not desired), the following lot size can be recommended: $LOT = AOD \times RLT$. Here RLT is the total replenishment lead and AOD is the average daily demand.

Case 2: Minimum increments
In this case, the lot size has to be an integer multiple of a minimum number ($QUNIT$). The lot size is given by $LOT = M \times QUNIT$. Here M is the smallest integer for which $M \times QUNIT$ is greater than $ROP - IP$, and IP is the inventory position.

Case 3: Min lot size
Lot size can be any number, but it must be above a minimum. That is, $LOT = \max(QMIN, ROP - IP)$. Here, $QMIN$ is the minimum lot size and IP is the inventory position.

Case 4: Max lot size
Lot size can be any number, but it must be below a maximum. That is, $LOT = \min(QMAX, ROP - IP)$. Here $QMAX$ is the maximum lot size and IP is the inventory position.

Case 5: Economic order quantity
When costs are available, an economic order quantity (EOQ) can be calculated. The cost inputs required for this calculation are $OCOST$ (fixed order cost paid per supply order) and $HCOST$ (inventory holding cost, dollars per unit held per day). The lot size is given by $LOT = \text{sqrt}(2AOD \times OCOST/HCOST)$.

Step 6: Calculate safety factor
When we previously described six different cases, we showed how to calculate key problem parameters in each case. Now we show how to calculate safety factors. In the inventory literature, these safety factors are calculated using functions such as the inverse of the standard normal distribution and the inverse of what is called g-function. These functions do not exist in standard query languages. Here we show the use of approximations to these functions; these approximations can easily be coded in query languages.

Cases 1 through 5
Safety factor k is given by the following:

$$k = -MULTIPLIER \times (0.5/A_2) \times \left[A_1 - \text{sqrt}\left(\max\{0, A_1^2 + 4A_2 \ln[\max(PVALUE, 1 - PVALUE) - 1]\}\right)\right],$$

where $MULTIPLIER = -1$ if $PVALUE < 0.5$, $MULTIPLIER = 1$ if $PVALUE \geq 0$, $A_1 = 1.363471$, and $A_2 = 0.266705$. The formula we provide here for the k value is an approximation for the inverse of the standard normal distribution function. Since some versions of SQL do not have the inverse normal as a built-in function, this formula is useful, since it uses mathematical function available in SQL. The formula is very accurate for practical purposes.

We tested the values of k in the interval [–3, 3] and found that the absolute maximum and absolute minimum errors are 0.035711 and –0.035711, respectively.

Case 6: Achieve service target (fill rate)
Safety factor k is given by the following:

$$k = \frac{A_0 + A_1 Z + A_2 Z^2 + A_3 Z^3}{B_0 + B_1 Z + B_2 Z^2 + B_3 Z^3 + B_4 Z^4},$$

where

$GVALUE = (1 - PVALUE) \times LOT/SDLT$
if there is a fixed lot size, LOT,

$GVALUE = (1 - PVALUE) \times ADLT/(RLT \times SDLT)$
if lot size is flexible,

$Z = \text{sqrt}\{2 \ln[\max(1, 5/GVALUE)]\}$,

and $A_0 = -5.3925569$, $A_1 = 5.6211054$, $A_2 = -3.883683$, $A_3 = 1.0897299$, $B_0 = 1.0000000$, $B_1 = -0.72496485$, $B_2 = 0.507326622$, $B_3 = 0.066913687$, and $B_4 = -0.003291291$. The derivation of these formulas can be found in [6]. These formulas are used for continuous review problems in which demand is also continuous. In this case, a simple (s, Q) inventory policy can be used if no more than a single replenishment order is outstanding in the supply pipeline. Here, Q is the lot size (LOT), and s is the reorder point (ROP). According to this policy, one places a replenishment order of Q when inventory position IP drops to s.

The above approximation is a very accurate estimation of safety factor k for normal demand. We tested the values of k in the interval [–3, 3] and found that the maximum and minimum errors are 0.000295 and –0.000205, respectively.

Step 7: Calculate optimal inventory policy
Inventory policy parameters include safety stock (SS), reorder point (ROP), and maximum inventory level (MAX). These parameters were explained earlier; their formulas, based on the safety factor and lot size (LOT), are $SS = k \times SDLT$, $ROP = ADLT + SS$, and $MAX = ROP + LOT$.

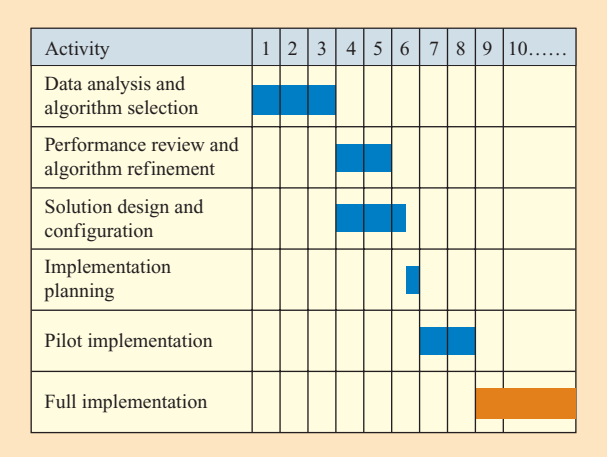

Figure 3

Typical plan for the SQL-based inventory optimization solution implementation. The numbers at the top represent weeks, from week 1 to week 10 and beyond.

Implementation issues

When implementing our approach, one must test the calculations with respect to actual data, since many of the assumptions may not be appropriate for the data. Simulation tests can be performed using actual sales and lead time data to observe how the inventory policies perform. If the performance is poor because of the normality of demand assumption, other distributions (such as gamma or log-normal) can be used. Approximations similar to what we presented for the normal distribution can be developed for other demand distributions. Alternatively, empirical distributions can be derived on the basis of sales data, and their percentiles can be used to calculate the safety stocks.

After achieving satisfactory performance results, a solution can be designed with a configuration that addresses the customer's requirements. A quick pilot implementation can follow and policies can be monitored in action for a subset of products before full implementation. A project plan for a simple implementation is given in **Figure 3.** The durations of activities are our estimates for a standard implementation. Actual implementation times can vary depending on the complexity of a customer's data systems and requirements.

Some of the algorithms we have presented are tested using actual historical data from clients. The simulation tests based on actual transaction data showed that for a variety of demand patterns, the formulas used in this paper are able to deliver target service levels on the average, although for some SKUs actual service can be below the target and for some it can be above. Most of the deviations from target service levels were within acceptable statistical tolerance levels in our test cases. However, caution must be used in cases in which the assumptions we made here are arbitrary. In using the formulas, we have shown a few issues that must be kept in mind; we reiterate some of these issues below.

Our basic assumption on which the inventory policy calculations are based is that historical demand is a good representation of future demand. If trends in the demand data exist, statistical forecasting models can be used to estimate demand during lead times. Such forecasting models can generate the mean daily demand (i.e., the daily demand forecast) and the standard deviation of daily demand. This forecast can replace the demand statistics calculated in Step 1. It is possible to program in SQL simple but effective forecasting techniques such as exponential smoothing and moving average, since these have simple forms.

The formulas normally work better with products that have a high demand volume (e.g., products that receive frequent orders). For low-demand products, inventory policies are very sensitive to the amount of individual orders; therefore, the actual service levels may vary significantly for such products. For some low-demand products, if it is critical to achieve desired inventory availability levels, it would help to increase the lot sizes to cover several months of demand.

Updating the inventory policy parameter (i.e., safety stock, reorder point, and lot size) calculations may or may not be desirable. If the demand pattern changes very little, the inventory policy should also not be expected to change much. In steady demand situations, monthly or quarterly parameter updates are reasonable. Otherwise, more frequent updates are needed.

Inventory replenishment orders should be triggered by observing the relative value of inventory position *IP* to the reorder point *ROP*. That is, when *IP* drops below *ROP*, an inventory replenishment order should be triggered for the predetermined lot size. If the reviews are periodic and the lot size is flexible, an order is placed in every period so as to bring the inventory position to ROP. This is also known as a *basestock* policy.

We note that IBM Global Business Services, IBM Research, and NIBCO (Northern Indiana Brass Company) have jointly implemented a version of the approach described in this paper to provide NIBCO a cost-effective inventory management solution. NIBCO has observed significant improvements in both inventory levels and on-time shipment performance since 2005.

Conclusion

Advantages

We have briefly explained our approach to create a virtually seamless inventory optimization solution

for database systems that use SQL. This approach has a number of potential advantages from both technical and business process perspectives. Here, we briefly explain some of these advantages.

Our approach simplifies optimal inventory policy calculations in such a way that existing databases and basic SQL commands can be used to rapidly code these calculations. Data-extraction templates and reporting templates of databases can be used to provide a fully functional solution. Hence, for simple inventory problems, no need exists for additional software.

Because this is a solution provided on an existing database system, there is no need to create new security protocols, network integration infrastructure, and Web access protocols. All of these elements are typically provided by the existing database system infrastructure for the users.

By using query capabilities of the existing database application, various reports of optimal policies, key performance indicator (KPI) projections and "what-if" analyses for decision support can be created. Many of these basic analyses can be done using mathematical functions available in SQL.

Because companies that use database applications already have trained staff, much less technical training is required in order to use this solution. As for the business process, the solution does not necessarily require changing existing inventory planning processes. It requires that the execution of recommended inventory policies be done properly by monitoring key metrics such as inventory position and sales orders.

Because this solution requires no new software, typical purchase costs and user fees do not apply. The maintenance costs can be controlled because this solution can be implemented as a part of the existing database/data-warehouse maintenance program.

Because the calculations are done at runtime through query commands, inventory managers and analysts can perform real-time what-if analysis using query reports. For instance, analysts can observe how inventory levels are affected if they change some key factors such as customer service objectives, operating objectives, and inventory policy types.

More complex inventory problems

Although the approach we have presented in this paper can solve some common inventory problems, many other cases exist for which more complex algorithms are needed. These can be coded following the standard approach and using a common programming language such as C or C++ or Java. Integration of these algorithms into the solution is typically done at the data extraction level. The results are then stored in database tables.

Since sales data is accessed and other historical demand statistics are calculated during data extraction from the transaction system, if advanced forecasting algorithms are desired in order to perform more accurate forecasts, these algorithms can be called during the data extraction process. The ability to use C or Java at the data extraction level gives enough flexibility to code advanced algorithms.

Figure 4 shows details of how this flexible design may work. As the figure indicates, the optimal policy calculations are performed during data extraction from the ERP system. The optimal inventory policies are stored, along with input data, and are made ready for reporting purposes. In addition, some metrics such as projected inventory levels, projected backlogs, and their confidence bands can also be calculated and stored. Queries are used for reporting only, and no optimization calculation is performed at the query level. This architecture provides the flexibility to place complex algorithms during the data extraction. Programming languages such as Java, C, or C++ can be used to code the algorithms.

This solution design has a drawback in that it may not be possible to obtain quick real-time results because of architectural and algorithmic complexity. However, potential scenarios can be analyzed in anticipation of user needs, and results can be stored prior to user requests. Thus, real-time response can be provided to anticipate queries.

Numerous sources of complications exist in inventory problems, and this is why the inventory-control literature is so rich. The multi-echelon nature of supply chain systems, substitutability of products, nonstationary nature of demand, products with bills of materials, and distribution systems are examples of the elements that make inventory problems complex and variable. (*Distribution systems* are systems in which inventory flows from manufacturing to central distribution centers, then to warehouses, and then to local stores.)

Since the literature is extensive, we can mention only a few key relevant publications. Readers who wish to become more familiar with these complexities may consult [10, 11, 19, 22–30] for multi-echelon systems. More specifically, for one-warehouse, multi-retailer systems, see [5, 14, 15, 31–36]. For assembly manufacturing systems, see [4, 30–35, 37].

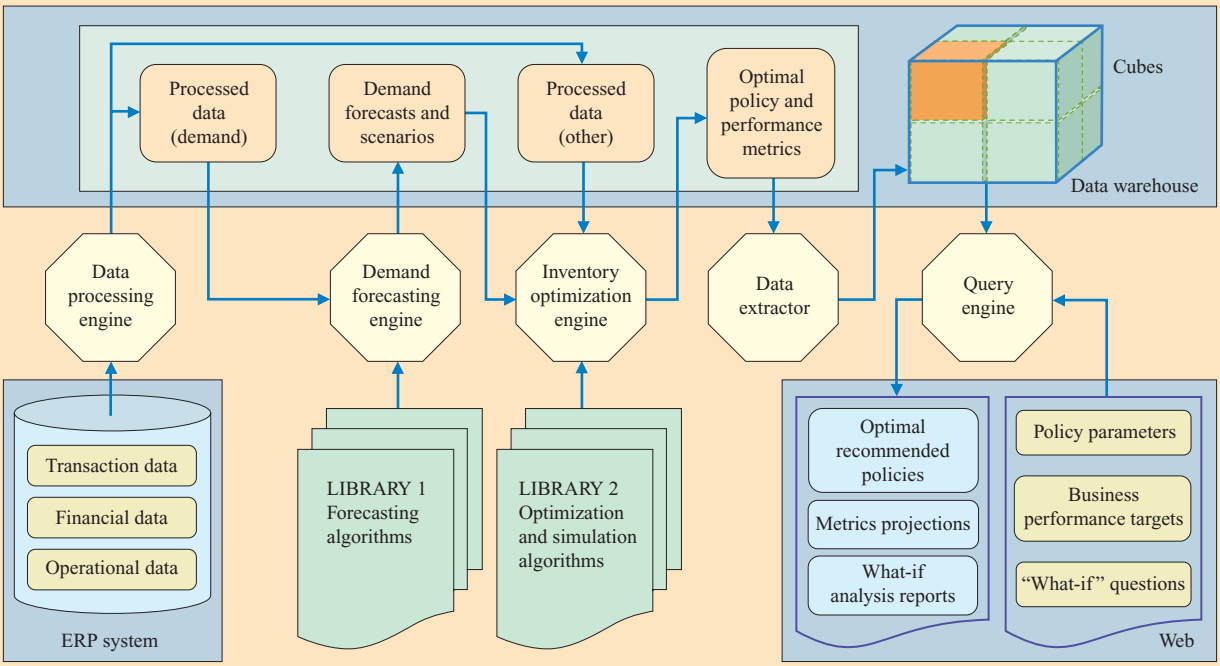

Figure 4

Flexible architecture that can support use of complex optimization algorithms. Within the "Web" box, yellow boxes represent user inputs, and light blue boxes represent outputs to the user.

References

1. S. Graves, A. R. Kan, and P. Zipkin, Editors, *Logistics of Production and Inventory*, Vol. 4, Elsevier (North-Holland), Amsterdam, The Netherlands, 1993.
2. A. C. Hax and D. Candea, *Production and Inventory Management*, Prentice-Hall, Upper Saddle River, NJ, 1984.
3. F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, Holden-Day, San Francisco, CA, 1980.
4. S. Nahmias, *Production and Operations Analysis*, Fourth edition, McGraw-Hill, New York, 2001.
5. L. B. Schwarz, Editor, *Multi-Level Production/Inventory Control Systems: Theory and Practice*, North-Holland, Amsterdam, The Netherlands, 1981.
6. E. A. Silver, D. F. Pyke, and R. Peterson, *Inventory Management and Production Planning and Scheduling*, John Wiley & Sons, Third edition, Hoboken, NJ, 1998.
7. R. J. Tersine, *Production and Operations Management*, North-Holland, New York, 1980.
8. P. Zipkin, *Foundations of Inventory Management*, McGraw-Hill, New York, 2000.
9. G. Hadley and T. M. Whitin, *Analysis of Inventory Systems*, Prentice-Hall, New York, 1963.
10. K. J. Arrow, S. Karlin, and H. Scarf, *Mathematical Theory of Inventory and Production*, Stanford University Press, Stanford, CA, 1958.
11. A. J. Clark and H. Scarf, "Optimal Policies for a Multi-Echelon Inventory Problem," *Manage. Sci.* **6**, No. 4, 475–490 (1960).
12. D. L. Iglehart, "The Dynamic Inventory Problem with Unknown Distribution of Demand," *Manage. Sci.* **10**, 429–440 (1964).
13. E. Ignall and A. Veinott, "Optimality of Myopic Inventory Policies for Several Substitute Products," *Manage. Sci.* **15**, No. 5, 284–304 (1969).
14. J. A. Muckstadt, "A Model for a Multi-Item, Multi-Echelon, Multi-Indenture Inventory System," *Manage. Sci.* **20**, No. 4, 472–481 (1973).
15. H. Scarf, "The Optimality of (*s*, *S*) Policies in the Dynamic Inventory Problem," *Mathematical Methods in The Social Sciences*, Stanford University Press, Stanford, CA, 1959, pp. 196–202.
16. D. Topkis, "Optimal Ordering and Rationing Policies in a Non-Stationary, Dynamic Inventory Model with *n* Demand Classes," *Manage. Sci.* **15**, No. 3, 160–176 (1968).
17. R. Ehrhardt, "Power Approximation for Computing (*s*, *S*) Inventory Policies," *Manage. Sci.* **25**, No. 8, 777–786 (1979).
18. J. R. Freeland and E. L. Porteus, "Evaluating the Effectiveness of a New Method for Computing Approximately Optimal (*s*, *S*) Inventory Policies," *Oper. Res.* **28**, No. 2, 353–366 (1980).
19. Y. S. Zheng and A. Federgruen, "Finding Optimal (*s*, *S*) Policies Is About as Simple as Evaluating a Single Policy," *Oper. Res.* **39**, No. 4, 654–665 (1991).
20. J. R. Groff and P. N. Weinberg, *SQL: The Complete Reference*, Second edition, McGraw-Hill, New York, 2002.
21. J. S. Song and P. H. Zipkin, "Managing Inventory with the Prospect of Obsolescence," *Oper. Res.* **44**, No. 1, 215–222 (1990).
22. D. Atkins and D. Sun, "98% Effective Lot Sizing for Series Inventory Systems with Backlogging," *Oper. Res.* **43**, No. 2, 335–345 (1995).
23. A. J. Clark and H. Scarf, "Approximate Solutions to a Simple Multi-Echelon Inventory Problem," *Studies in Applied Probability and Management Science*, K. J. Arrow, S. Karlin,

and H. Scarf, Editors, Stanford University Press, Stanford, CA, 1962, pp. 88–110.
24. M. A. De Bodt and S. C. Grave, "Continuous Review Policies for a Multi-Echelon Inventory Problem with Stochastic Demand," *Manage. Sci.* **31**, 1286–1295 (1985).
25. A. Federgruen and P. Zipkin, "Computational Issues in an Infinite-Horizon Multiechelon Inventory Model," *Oper. Res.* **32**, No. 4, 818–836 (1984).
26. S. C. Graves and L. Schwarz, "Single Cycle Continuous Review Policies for Arborescent Production/Inventory Systems," *Manage. Sci.* **23**, No. 5, 529–540 (1977).
27. P. Jackson, W. Maxwell, and J. Muckstadt, "Determining Optimal Reorder Intervals in Capacitated Production–Distribution Systems," *Manage. Sci.* **34**, No. 8, 938–958 (1988).
28. L. B. Schwarz and L. Schrage, "Optimal and System Myopic Policies for Multi-Echelon Production/Inventory Assembly Systems," *Manage. Sci.* **21**, No. 11, 1285–1294 (1975).
29. C. C. Sherbrooke, *Optimal Inventory Modeling of Systems, Multi-Echelon Techniques*, John Wiley & Sons, Hackensack, NJ, 1992.
30. W. Zangwill, "Eliminating Inventory in a Series Facility," *Manage. Sci.* **33**, No. 9, 1150–1164 (1987).
31. S. Axsater, "Simple Solution Procedures for a Class of Two-Echelon Inventory Problems," *Oper. Res.* **38**, No. 1, 64–69 (1990).
32. S. Axsater, "Evaluation of Lot-Sizing Techniques," *Intl. J. Production Res.* **24**, No. 1, 51–57 (1986).
33. J. A. Muckstadt and R. O. Roundy, "Multi-Item, One-Warehouse, Multi-Retailer Distribution Systems," *Manage. Sci.* **33**, No. 12, 1613–1621 (1987).
34. K. Rosling, "Optimal Inventory Policies for Assembly Systems Under Random Demands," *Oper. Res.* **37**, No. 4, 565–579 (1989).
35. R. Roundy, "98%-Effective Integer-Ratio Lot-Sizing for One-Warehouse Multi-Retailer Systems," *Manage. Sci.* **31**, No. 11, 1416–1430 (1985).
36. L. B. Schwarz, B. Deuermeyer, and R. Badinelli, "Fill-Rate Optimization in a One-Warehouse *N*-Identical Retailer Distribution System," *Manage. Sci.* **31**, No. 4, 488–498 (1985).
37. C. C. Sherbrooke, "VARI-METRIC: Improved Approximations for Multi-Indenture, Multi-Echelon Availability Models," *Oper. Res.* **34**, No. 2, 311–319 (1986).

Received September 25, 2006; accepted for publication October 16, 2006; Internet publication May 25, 2007

Kaan Katircioglu *IBM Research Division, Thomas J. Watson Research Center, P.O. Box 218, Yorktown Heights, New York 10598 (kaan@us.ibm.com)*. Dr. Katircioglu is a leading researcher and a consultant in supply chain management at the IBM Thomas J. Watson Research Center. He has more than ten years of experience in the field of operations research, management science, and logistics. His expertise covers the areas of inventory optimization, distribution and manufacturing operations management and planning, and e-business and supply chain management. After completing his Ph.D. studies in management science, in 1996 he joined the IBM Thomas J. Watson Research Center. Since then, Dr. Katircioglu has worked on several projects for various divisions of IBM and its customers. He is the inventor of the Express Services for Inventory Management offering announced by IBM in July 2005. Dr. Katircioglu has published numerous papers, made several conference presentations, and received patents for his work. He is a member of INFORMS and IEEE.

Timothy M. (Tim) Brown *IBM Global Business Services, 4111 Northside Parkway NW, Atlanta, Georgia 30327 (tmbrown@us.ibm.com)*. Mr. Brown is an Associate Partner with the Global Business Services (GBS) division of IBM Global Services. Within GBS, Mr. Brown currently focuses his efforts on defining and implementing supply chain transformation approaches for midsized companies. He is a certified business transformation consultant in the IBM Supply Chain Management competency, with 25 years of experience in logistics management, supply chain strategy, and supply chain operations improvement. He is also a certified Project Management Professional and is certified in production and inventory management. Mr. Brown has worked with clients of all sizes and numerous industry sectors. He received an M.B.A. degree from Georgia State University and a B.S. degree in management science from the Georgia Institute of Technology.

Mateen Asghar *IBM Global Business Services, One Rogers Street, Lotus Development, Cambridge, Massachusetts 02142 (mateen.asghar@us.ibm.com)*. Mr. Asghar is an Industry Solution Manager at IBM Global Business Services (GBS). He is responsible for the SAP Business Intelligence Express solution and is a subject matter expert on the cross-solution integration for small and medium-sized companies in various industries. He has more than ten years of experience with SAP R/3 systems as a technical development leader, and he is a Business Intelligence subject matter expert, an integration subject matter expert, and a leader for various large-scale and complex projects. Mr. Asghar has participated in several projects for Fortune 500 companies, primarily in the area of consumer products, aerospace, and defense industries. His expertise is in managed full-scale SAP systems development projects with large teams utilizing onsite/offshore models. He was the lead architect of the Express Services for Inventory Management offering announced by IBM in July 2005, and he designed the complete solution utilizing various components of SAP, ERP, and BI.