Informix SOFTWARE

Informix-NAG Financial DataBlade Module

Abstract

The Informix®-NAG DataBlade® module adds analytical capabilities to the Informix database server. This has a number of important advantages. Analysis generally reduces the amount of data, sometimes down to a single figure, so the sooner this analysis takes place the better. It is the result that is needed, not the raw data. The Informix-NAG DataBlade module moves the analysis close to the data. Doing the analysis in the database server also means that the identical analysis is available to disparate client processes. The client process (C program, ODBC or JDBC connect programs, Web browser) can use the same server implemented functions. Consider a simple example of calculating the variance-covariance matrix for the 100 shares over 1 year using the daily closing prices. The ratio of input data to output data is 5-to-1. If the variance-covariance matrix can be produced in the server, then only this much smaller result needs to be sent to the client. Since the client can be any program, all the client programs can now share the same analytical functions.

The numerical analysis is performed by functions from the standard NAG Fortran libraries executed directly by the server process. This gives the best possible performance by the highest quality numerical functions; you get speed and mathematical integrity.

The user can now control the analysis with User Defined Routines (UDRs). These routines can be written in C, Stored Procedure Language (SPL), Java etc. The mathematically intense processing can be left to the NAG Fortran functions accessed via the Informix-NAG DataBlade module.

The Informix-NAG DataBlade module is a natural partner of the Informix TimeSeries DataBlade module. The TimeSeries DataBlade module efficiently stores and retrieves time stamped data, while the Informix-NAG DataBlade module has the powerful analytical capabilities for extracting information from the mountain of raw data.

The accompanying white paper explores the advantages of the Informix-NAG DataBlade module further.

The list of NAG functions currently supported is included, and this will be expanded to meet customer requirements.

Table of Contents

Executive Overview

This paper describes the new Informix-NAG DataBlade module, the module that makes it possible to do complex analysis accurately and quickly in the database server. These accurate, quick, and flexible analytical capabilities are the benefits of the logical step of moving the analysis much closer to the data and not continually moving the data to the analysis.

This paper contains a brief description of the current situation, the problems, and the solutions provided by the Informix-NAG DataBlade module. The paper finishes with some simple examples.

Situation

Increasing amounts of data

The costs of data collection and storage have declined rapidly over the years and are expected to continue to fall. This has lead to ever-increasing amounts of data being stored covering many areas of every day life and business activity. There have been a number of factors reducing the cost of data collection. In the early days of computing, data was typed in by intermediaries, this is slow, expensive, and error prone. Much data is now scanned, generated automatically, or typed in by the consumer, which is faster, cheaper and more accurate.

This easy data collection, coupled with low-cost data storage, faster access, and cheaper CPU power, mean that all the "raw materials" for turning the data into competitive business are available. All that is needed now is an efficient implementation of the correct analysis.

What to do with the data

There is no advantage in treating data storage as write only memory. Data only becomes valuable when it is turned into a business advantage. This advantage can come from understanding why things have happened, and more importantly, to calculate the probabilities events that will happen.

Analysis

The analysis of the data turns historical information into future profit. Simple questions like

- How much risk is there in a share portfolio?
- How much product should be made tomorrow?
- When should more capacity be added to the network?

have simple answers. The answers may be simple, but they only reveal themselves after transforming large amounts of data using complex numerical analysis. This analysis is becoming more complex as the sources and complexity of data increases and the cost of CPU power decreases. There is now a need for more frequent analysis of the data as more frequently updated data becomes easily available.

The results of the analysis need to be presented in ways that can be used by other processes or understood and interpreted by humans. The best and quickest analysis is of no use if it can't be used or understood.

Problem

Turing data into strategic and tactical business information is a two-stage process. Analytical techniques need to be evaluated and proven. Once proven, they can be used with confidence when making business decisions.

Investigative and testing phase

In the investigative and testing phase, there is a tight loop involving analysis, evaluation of the results, and refinement of the analysis.

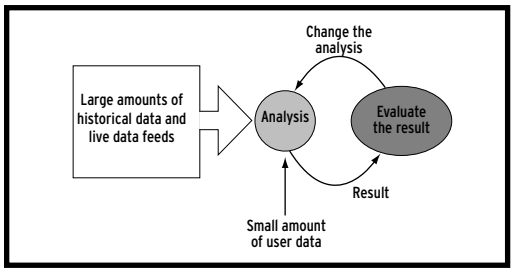

Figure 1 Figure-Text

The investigative phase of creating an analytical solution requires flexibility, the flexibility to quickly access data in different ways and to analyse the data using different techniques. The accuracy of the results and confidence in that accuracy are crucial in making the correct design decisions. Data presentation is important in understanding what is happening.

Production phase

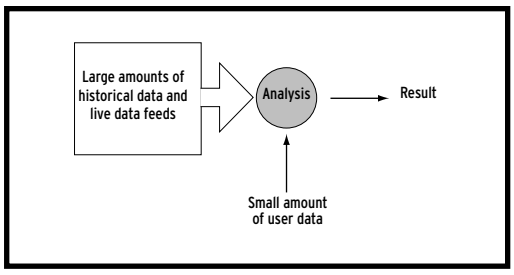

Figure 2 Figure-Text

The main factors in production systems are the accuracy of (and confidence in) the results and the efficiency of getting these results. Good business decision can only be made with correct results extracted from the mountain of data, getting the result efficiently means that it costs less to get the result, or getting the result earlier, or both.

It is important to have consistency across all processes using the derived information. This means either using the same algorithms in all client processes, which may be difficult, if not impossible, with many different client programs and languages, or saving intermediate analytical results in the server.

Traditional RDBMS architecture for complex analysis

The standard RBMS solution is to move much of the data from the database through the server to the client process for analysis, and in many cases storing intermediate data back in the database.

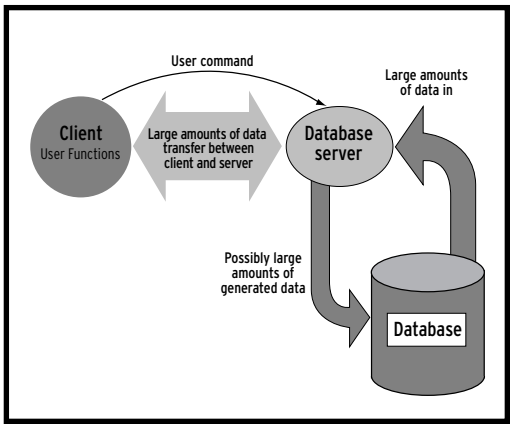

Figure 3 Figure-Text

Summary

Devising new analytical procedures needs flexible data access and analytical procedures. The results need to be accessible to a wide range of programs that can manipulate the results in ways that help with the interpretation of the results and the evaluation of the suitability of the analytical techniques.

Speed can be important in developing new analytical routines and is often crucial in production systems either to get the results quickly or to reduce the cost of getting the result in a given time.

Accuracy, and confidence in the accuracy, are essential in both investigative and production systems. If the results are inaccurate, or you can not be certain they are accurate, then there is little point in analysing the data in the first place.

Solution

The object relational technology in Informix Dynamic Server.2000™ can play a major part in solving the problems of the complex analysis of the mountain of data. The ORDBMS is extensible, i.e. new data types and functions can be added to the database server and these types and functions can be used in SQL statements and in the Stored Procedure Language (SPL). The functions themselves can be written in SPL, C, or Java.

The new data types introduced by the Informix- NAG DataBlade module are vectors and matrices. These types work closely with the existing TimeSeries DataBlade module to provide suitable numerical data types for analytical work. These types are used by the Informix interface to the highly regarded analytical functions from The Numerical Algorithms Group (NAG) see http://www.nag.com for more details.

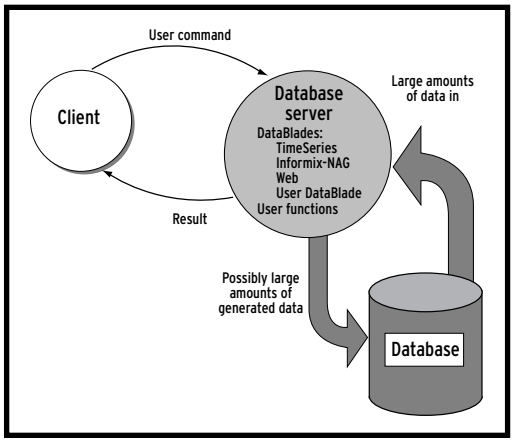

Figure 4 Figure-Text

Accuracy

The problems of writing and testing complex numerical and statistical functions are elegantly solved by linking in functions from NAG. This NAG connection has made available some of the most popular NAG functions in the database server moving the mathematical analysis very close to the data. The NAG algorithms have been implemented by mathematicians who understand numerical analysis. NAG have many thousands of users relying on their 1,000+ functions.

Flexibility

The flexibility of the solution derives from the ability to use the Informix- NAG DataBlade module functions in SPL and user written C functions. SPL is an interpreted language, which makes it very quick and easy to change. SPL functions and procedures can be called from higher level SPL functions and procedures and from SQL. The SQL statements and SPL functions and procedures can be invoked from client programs (SQL editors, Web browsers, and any application capable of making ODBC calls, Java via JDBC calls). Further flexibility and performance can be gained from writing server functions in C or Java.

Performance

There are many performance benefits inherent in this solution. The TimeSeries DataBlade module is a natural partner to the Informix-NAG DataBlade module. Much data is time based and the TimeSeries DataBlade module provides a fast and powerful way to store and access this time-based data.

The new data-types (vector and matrix) in the Informix- NAG DataBlade module provide efficient ways of operating on large amounts of data and quick and easy storage as "smart large objects." The vector and matrix arithmetic used by mathematicians can be quickly transformed in to analytical SPL functions.

The NAG functions are accurate and very fast, this is NAG's forte, and the Informix-NAG DataBlade module makes these functions available in the database server. There is no passing of the data between processes and no context switching, the functions have been moved close to the data so there is no need to move the data to functions in a client program.

Functionality

The Informix-NAG DataBlade module contains two new data types:

- Vector and matrix data types and their support functions

Many of the most popular functions have been included in the NAG DataBlade module with more to follow. The areas currently include functions from the following NAG libraries:

- Basic Linear Algebra Subroutines (BLAS)
- Roots of One or More Trancendental Equations
- Interpolation
- Curve and Surface Fitting
- Matrix Factorizations
- Eigenvalues and Eigenvectors
- Simple Calculations on Statistical Data
- Correlation and Regression Analysis
- Multivariate Methods
- Random Number Generators
- Smoothing in Statistics
- Time Series Analysis

with more to follow. Some business specific function have also been included. Full descriptions of the types and functions is available at http://www.informix.com/uk/nag.

Server solution

This is a server solution. The results of these functions are available to all client processes providing a common server solution for all client processes. The client process could be an SQL editor, ESQL/C program, any program supporting ODBC or JDBC, and Web browsers. The server solution is particularly important in thin client architectures where getting data to the client may be slow, and even the simplest numerical analysis in the client will be very slow.

Examples

This section uses simple examples from finance, retail, and telecommunications to illustrate the flexibility, performance, and functionality of the Informix-NAG DataBlade module solution.

Internal Rate of Return

This simple financial example is the calculation of the present value of a stream of future payments and illustrates the flexibility of using the Informix-NAG DataBlade module. The compounding of the interest can be treated in two ways, continuous or discrete. The Informix-NAG Finance DataBlade module has functions to calculate the present value both ways:

value = PVCont(rate, amounts, times)

and:

value = PVDisc(rate, amounts, times)

where

value = The present value
rate = Interest rate per unit time
amounts = Vector of payments
times = Vector of times of these payments

A closely related calculation is the internal rate of return (IRR) of an investment. There is no function to calculate the IRR, but it is easy to write one using SPL that calls the present value functions to calculate the IRR.

The SPL code to calculate the IRR is:

```
--
-- An SPL function to calculate the
   Internal Rate of Return for a stream
-- of payments   irr = IRR(amounts, times)
--
-- Inputs: amounts  - Vector of cash flows
--       times    - Vector times for the cash
                    flows
-- Return:irr    - The internal rate of
                    return per unit time
-- Exceptions: 100        - The IRR could be
                            found
--
create function IRR(amounts vecDblType,
times vecDblType)returning float;
define params vecDblType;
define rate, f, float;
define ifail, loop_count int;

let loop_count = 50;
-- Set up the input vector to udrc05axf()
-- params = {initial_guess_at_rate, ind};
-- see documentation for full description

let params = '.01, 1';
-- Loop round until a solution is found
   ((getNth(params, 1) == 0),
-- for a maximum of 50 iterations
while ( loop_count > 0 and getNth(params,
1) != 0 )

-- Get the present value for this cashflow
   at this rate
let f = PVDisc(getNth(a, 0), amounts,
times);

-- Get the next value of rate to try
let ifail = udrc05axf(a, f, 0.001, 0, 0.1e-
10);

-- Decrement loop count
let loop_count = loop_count - 1;

end while;

-- If ifail isnt = 0 or loop_count >= 50
   then there was an error
if (ifail != 0 or loop_count == 0 ) then
 raise exception 100, 0, "IRR not found";
end if;

-- Return the IRR
 return getnth(params, 0);
end function;
```

This code is available on the Informix-NAG Web page. The SPL code uses the NAG function c05axf() that calculates the "Zero of continuous function by continuation method, from given starting value (reverse communication)." This IRR function could easily be changed to use the continuous discounting present value function, just change PVDisc() to PVCont(). The type of input parameters can be changed to TimeSeries and the amounts and times extracted from the TimeSeries. The method of calculation of the IRR could be changed by using the NAG function c05azf() that calculates "Zero in given interval of a continuous function by Bus and Dekker algorithm (reverse communication)." The function can be made more robust by checking for multiple zeros of the function.

This illustrates the flexibility of the ORDBS approach. SPL is a very good language for controlling functions such as the IRR calculation. The SPL can easily and quickly be modified and tested. The NAG functions do the mathematical work.

Once the IRR() function has been changed to suit the application, the SPL function can be used in SQL statements, called from client programs via ODBC, JDBC, ESQL/C, ESQL/Cobol etc. The function can be used in functional indices, e.g, if the function was changed to work from a time series column representing cash-flow (cash_flow) in a table of investments, then the IRR of the investments can be indexed:

```
  create index example_index on
investments(IRR(cash_flow));
```

This allows fast, indexed queries, on the investment table based on the IRR. The maintenance of the index during insert, update, and delete of the investment table is transparent to the user and applications.

Share prices

Share price information is a classic example of time series data, and the TimeSeries DataBlade module is a natural and efficient way to store the data. The tests in this section are based on a years daily share data, approximately 250 prices, for a number of shares.

A simple analytical operation might be to:

- Get the price information
- Convert the prices to returns
- Produce a variance-covariance matrix on the returns
- Save the variance-covariance matrix
- Retrieve the variance-covariance matrix

This sequence of operations can be performed on any number of shares. If this analysis is performed on n shares, then there are 250 x n data items that need to be extracted from the database and converted to returns. This results in an n x 249 matrix of stock returns. This matrix is converted to an n x n symmetrical variance-covariance matrix, which can be stored as n(n+1)/2 data elements.

Traditional solution

The traditional solution is to do the analysis in a client process, the steps are:

- Select the prices
- Do the analysis
- Insert the result into a table

this stored results can then be used by other client processes by:

- Selecting (from the client) the elements of the variance-covariance matrix

The problems are the amount of data that needs to be stored, selected, and shifted between the server and client.

ORDBS solution

An efficient ORDBS solution can be designed using the TimeSeries and Informix-NAG DataBlade modules. The storage of the stock price data is efficiently handled by the TimeSeries DataBlade module. The steps in the ORDBS solution are:

- Select the share prices (held as TimeSeries)
- Convert to vectors of returns, insert into a matrix, analyse using a NAG function
- Store as a smart large object

This stored smart large object can then be used by other functions by:

- Selecting the smart large object in the server

This is much more efficient because less data is extracted from, and inserted into, the database and all the processing is performed in the server. The only data passed between the client and server is the instruction to execute the procedure and any parameters the procedure may need.

Timings

A simple test was performed where a variance-covariance matrix of stock returns was calculated for different numbers of shares. The times for the separate stages for the traditional and ORDBMS solutions outlined above were recorded.

The *relative* times for different numbers of shares are shown in Table 1

	Number of shares					
	10	**30**	**100**	**300**	**1000**	**3000**
Traditional solution						
Retrieve price data	2	7	22	64	263	789
Do the analysis	0	0	0	3	107	1421
Save results	1	1	5	36	629	4630
Total	**3**	**8**	**27**	**103**	**1000**	**6840**
Retrieve results	**1**	**1**	**2**	**13**	**144**	**1280**
ORDBS solution						
Retrieve price data	1	2	5	15	68	226
Do the analysis	0	0	0	3	107	1421
Save results	1	1	1	2	8	122
Total	**2**	**3**	**6**	**20**	**183**	**1769**
Retrieve results	**1**	**1**	**1**	**4**	**9**	**51**

Table 1
The ORDBS solution is up to 5-times faster than the traditional solution and selecting the results for use in further analysis is up to 25-times faster.

The full test code is available through the Informix-NAG Web page. The fragment of SPL code that extracts the share price data and converts it to a vector of returns is:

```
let i = 0;
--tsToReturnsVec() converts the TimeSeries
--of prices to a vector of returns
  foreach
   select
tsToReturnsVec(share_val,1,'A','A','A',NULL,
NULL)
    into tmpVec
    from shares
    where share_id >= ip1
    and share_id <= ip2

-- Insert vector into matrix
let cc = updateRow(returnsMatrix, t, i);

 let i = i + 1;
end foreach;
```

The fragment of SPL code that calls the NAG function g02baxf() that "Computes (optionally weighted) correlation and covariance matrices" is:

```
let cc = udrg02bxf('U', returnsMatrix,
NULL::vecDblType,
        xbar, std, varCovar, r);
```

This function also returns the mean of the price for each share, `xbar`, standard deviation, `std`, and the correlation matrix , `r`.

These examples shows not only the flexibility of the solution but how much faster it can be than a traditional RDBMS implementation.

Retail predictions

An example from retail is how many factors (e.g. weather, advertising, school holidays) can affect sales of an item. A technique called multiple linear regression can be used to link the affected (dependent) variable, in this case sales of an item, with the causes (independent variables).

In this example both the cause and effect can be stored as time series. If 20 independent variables were used with daily values for each for approximately 3 months, then the input data consists of:

1 x 100 values for the dependent variable

20 x 100 values for the independent variables

and the result is

20 values that express how important the independent variables are

20 values showing how the dependant variable is linked to the independent variable

In the traditional solution 2,100 data items need to be sent to the client compared with just the result of 40 data items in the ORDBS solution. This factor of over 50 difference is crucial if there are 1,000s of items that need to be analysed separately for 100s of locations.

The Informix-NAG function for multiple linear regression, g02daf() that "Fits a general (multiple) linear regression model" can be used for this type of analysis. The fragment of SPL code that performs the multiple linear regression analysis is:

```
let ifail = udrg02daf('M', 'W', weatherMat,
isx, salesVec,
        weights, params, b, se, cov, res,
        h, q, p, tol);
```

where the important parameters in this example are:

'M'	= A mean term needs to be entered
'W'	= Weighted least square fit needs to be used
weatherMat	= A matrix of weather details (or other independent variables
isx	= Which independent variable need to be used
salesVec	= The vector of sales (or dependent variable)
weights	= Vector of weights (which days to include)
params	= Vector of various output parameters including the degrees of freedom, residual sum of squares
b	= Vector of least squares estimates of the parameters of the model
se	= Vector of standard errors in the least-squares estimates

The Informix-NAG Web page has a simple example with test data.

This example shows how the Informix-NAG DataBlade module can cut down the amount of data transferred by a factor of 50 and the type of complex analysis (in this case multiple linear regression) that can now be performed in the database server.

Telecommunications

A common telecommunications measure is the "busy hour traffic," which is (as the name suggests) the total traffic over the busiest hour of the day. This busy hour can move from one day to the next. If the traffic values are held in a regular time series column, then the function tsToPeakPeriod() function can transfer the regular readings to a vector of daily peak period values. If the network traffic is recorded every 5 minutes, then there are 288 (24 x 12) readings per day. This function returns a single peak period value from these 288 readings, just one data item per day. The traditional method is to return all the readings and calculate the peak values on the client, which means transferring 288 times as much data to the client.

The SQL statement to extract a vector of daily peak hour values from a time series column containing 5 minute readings is:

```
select tsToPeakPeriod('traffic_data', 1, 12)
from telco_table;
```

The same function can be used in SPL and to get the values between 2 dates, e.g.:

```
select tsToPeakPeriod(clip(traffic_data',
startDate, endDate), 1, 12)
  into busyHourVec
  from telco_table;
```

The vector of peak hour values can be analyzed further in the engine using SPL, e.g. weekly and day of week averages with standard deviations, traffic growth rates, and with the ErlangB() function, can be used to predict traffic (and revenue) loss due to congestion.

The Informix-NAG DataBlade module has these two simple telecommunications function, tsToPeakPeriod() and ErlangB(), which can be used in conjunction with the TimeSeries DataBlade module, Informix-NAG data-types and functions to construct complex analytical routines. This example is used in the telecommunications white paper, "Time Series: The Next Step for Telecommunications Data Management" http://www.informix.com/informix/whitepapers-datablade.

Summary

The Informix-NAG DataBlade module can be used to quickly implement complex server-side analysis using new data types and functions. The NAG routines in the DataBlade module are used and trusted by 1,000s of mathematicians worldwide and are highly regarded for their accuracy and speed. Using functions in the server moves the analysis close to the data instead of moving vast amounts of raw data to analytical functions in the client. The data is analyzed first and the results of the analysis either kept in the database or returned to the client. This can be many times more efficient and is particularly powerful when combined with "thin" clients. This is a flexible solution because the NAG functions can be called from SPL functions, which are quick to write and test and can easily be added (and removed) from a running database server.

Informix-NAG Finance DataBlade Module—Function List

The NAG functions that have been included version 1.10 of the DataBlade module are:

Function	Description
C05AVF	Binary search for interval containing zero of continuous function (reverse communication)
C05AXF	Zero of continuous function by continuation method, from given starting value (reverse communication)
C05AZF	Zero in given interval of continuous function by Bus and Dekker algorithm (reverse communication)
E01BAF	Determines a cubic-spline interpolant to a given set of data.
E02ADF	Least-squares curve fit, by polynomials, arbitrary data points
E02AEF	Evaluation of fitted polynomial in one variable from Chebyshev series form (simplified parameter list)
E02AFF	Least-squares polynomial fit, special data points (including interpolation)
E02BAF	Least-squares curve cubic spline fit (including interpolation)
F01CKF	Matrix multiplication (and vector-matrix, matrix-vector)
F01CRF	Matrix transposition
F01ZAF	Convert real matrix between packed triangular and square storage schemes
F02FAF	All eigenvalues and eigenvectors of real symmetric matrix (Black Box)
F06EAF	Dot product of two real vectors (SDOT/DDOT)
F06EDF	Multiply real vector by scalar (SSCAL/DSCAL)
F06FDF	Multiply real vector by scalar, preserving input vector
F06JLF	Index, real vector element with largest absolute value (ISAMAX/IDAMAX)
F07FDF	Cholesky factorization of real symmetric positive-definite matrix (SPOTRF/DPOTRF)
F07MDF	Bunch-Kaufman factorization of real symmetric indefinite matrix (SSYTRF/DSYTRF)
G01AAF	Mean, variance, skewness, kurtosis etc, one variable, from raw data
G01ALF	Computes a five-point summary (median, hinges and extremes)
G01DAF	Normal scores, accurate values
G01DBF	Normal scores, approximate values
G01EAF	Computes probabilities for the standard Normal distribution
G01EBF	Computes probabilities for Student's t-distribution
G01FAF	Computes deviates for the standard Normal distribution
G02BAF	Pearson product-moment correlation coefficients, all variables, no missing values

G02BBF	Pearson product-moment correlation coefficients, all variables, casewise treatment of missing values
G02BJF	Pearson product-moment correlation coefficients, subset of variables, pairwise treatment of missing values
G02BUF	Computes a weighted sum of squares matrix
G02BXF	Computes (optionally weighted) correlation and covariance matrices
G02DAF	Fits a general (multiple) linear regression model
G02DKF	Estimates and standard errors of parameters of a general linear regression model for given constraints
G02GBF	Fits a generalized linear model with binomial errors
G02HAF	Robust regression, standard M-estimates
G03AAF	Performs principal component analysis
G03CAF	Computes the maximum likelihood estimates of the parameters of a factor analysis model, factor loadings, communalities and residual correlations
G05CBF	Initialise random number generating routines to give repeatable sequence
G05CCF	Initialise random number generating routines to give non-repeatable sequence
G05DDF	Pseudo-random real numbers, Normal distribution
G05FDF	Generates a vector of random numbers from a Normal distribution
G10CAF	Compute smoothed data sequence using running median smoothers
G13AAF	Univariate time series, seasonal and non-seasonal differencing
G13ABF	Univariate time series, sample autocorrelation function
G13AUF	Computes quantities needed for range-mean or standard deviation-mean plot
G13BAF	Multivariate time series, filtering (pre-whitening) by an ARIMA model
G13CBF	Univariate time series, smoothed sample spectrum using spectral smoothing by the trapezium frequency (Daniell) window
G13DMF	Multivariate time series, sample cross-correlation or cross-covariance matrices
G13DNF	Multivariate time series, sample partial lag correlation matrices, chi-squared statistics and significance levels
M01CAF	Sort a vector, real numbers
M01DAF	Rank a vector, real numbers
S01BAF	ln (1+x) (3 version udrs01baf(double) returns double and udrs01baf(mat|vec) returns ifail)
X05AAF	Return date and time as an array of integers
X05BAF	Return the CPU time

There are 30+ vector functions and 30+ matrix functions

The DataBlade module includes some business related functions. This list will be increased to meet customer requirements.

Function	Description
tsToVec	Converts a time series to a vector. The user can define how missing values are handled.
tsToReturnsVec	Converts a time series of values into a vector of returns. The user can define how missing values are handled.
PVCont	Calculates the present value of future payments using continuous discounting
PVDisc	Calculates the present value of future payments using continuous discounting
ErlangB	Calculates the probability that blocking occurs for a given offered traffic and number of servers
invErlangB	Calculates the number of servers necessary to give a probability of blocking less than the specified value for some level of offered traffic
tsToPeakPeriod	Converts a regular time series of readings into a peak period value for the day
csv	A range of functions to output vectors and matrices to csv files

About Informix

Informix Software is the technology leader in software infrastructure solutions for the Internet—providing a fast, simple and complete way to bring businesses to the Web. Based in Menlo Park, Calif., Informix is the first and only company to integrate e-commerce and business intelligence on a true Internet infrastructure. The company's highly scalable Web engines, together with its personalized content management, real-time analytics and media asset management capabilities, offer customers a unique competitive advantage. For more information, contact the nearest sales office or visit the Web site at www.informix.com.

Informix®
SOFTWARE
way to web™

4100 Bohannon Drive
Menlo Park, CA 94025
Tel. 650.926.6300
www.informix.com

INFORMIX REGIONAL SALES OFFICES

Asia Pacific	65 298 1716	Japan	81 3 5562 4500
Canada (Toronto)	416 730 9009	Latin America	305 591 9592
Europe/Middle East/Africa	44 208 818 1000	North America	800 331 1763
Federal	703 847 2900		650 926 6300

Printed in U.S.A. 6/00
000-22130-70