Redbooks Paper

Victor Chao
Leticia Cruz
Nin Lei

# Local versus Remote Database Access: A Performance Test

When tuning a database for better performance, one area to examine is the proximity of the database to the application. Can the database be located on the same server as the application? Will a database server located far away from the application server cause performance degradation? To answer these questions, we conducted a series of measurements at the IBM zSeries Benchmark Center to determine performance impact based on the distance between the application and database servers. This paper discusses the results of our tests.

# Objectives

Determining where to host an application hinges on many factors, ranging from platform constraints to capacity availability. Certain applications run on a specific platform only, and that limits the placement selection. Another consideration is server capacity availability. Normally it is a good practice to co-locate the application and the database components. A local connection is more efficient and it eliminates network traffic. This reduces data access time and delivers good performance. However, it is not always possible to have sufficient capacity to house the application and the database on the same server. A compromise may be necessary to move the application to a different server. This arrangement restricts the application to use a remote data access mechanism. Clearly there is performance degradation with this approach. This paper provides measurement data quantifying the performance differences between local and remote accesses to a database. With this information, trade-offs can be made about locating an application locally versus remotely.

Some applications execute a large number of SQL statements but produce a single answer set. For example, a program looking up the rows of an employee table and performing business logic to produce demographical information. This program is a good candidate for stored procedures. We performed a set of measurements in this study with the objectives of comparing performance between call level interface (CLI) and stored procedure access of database contents. Results from this set of measurements assists programmers in making decisions of program migration to stored procedures.

We all know the distance between an application server and a database server can affect data access response times. Certainly a database server located tens of miles away from an application server would yield measurable and possibly noticeable performance degradation. But how about a database that is located in a separate building within a large business campus? In fact many debates were held in the technical communities to determine whether performance discrepancies between benchmarks with similar configurations were simply due to network distance differences.

Given the expenses of using a dedicated environment for a benchmark, it is quite common to use production/test systems to conduct a benchmark during off-shift hours. This presents a situation where the benchmarked components could be reasonably far apart. For instance, an application runs on a distributed platform server located in one building while the database product runs on a mainframe server installed in another building. To address this question of network latency, we ran measurements comparing access of a database only 150 feet away versus a database located about a mile away. Results from this experiment would tell us the impact of network distance to performance.

Although the tests conducted in this study were based on DB2® for z/OS®, the results could be applied to other database products. It was not the intent of this study to put significant stress to the database manager nor was it designed to use a comprehensive set of database manager services. Instead, only minimal DB2 services were used by the test program. The focus of the measurements was on evaluating the speed of data access to DB2 with local and remote programs.

# Workload

A C program was written to access DB2 data. It used the CLI protocol so that it could run locally and remotely without any modifications. All CLI statements executed dynamically. To minimize the overhead of rebinding, dynamic statement caching was enabled in DB2.

This program performed data accesses in a loop. It issued a SELECT statement against the SYSDUMMY table returning a result set of one row. No accesses to the system catalog or

any user tables were performed. This approach minimizes database processing with the deliberate outcome of network latency as the largest component of the program execution response time. A get-time function was performed before and after each call to DB2, with the difference between these two times yielding the end-to-end database access time (including network latency). The program then stepped through the loop one million times and computed a set of statistics, including average, total, and standard deviation.

For the stored procedure test cases, this program was broken up into a calling program and a stored procedure. Although CLI calls were possible in a stored procedure, the program was modified to use static SQL statements to adopt production practices. As a result, performance difference between static and dynamic SQL statements was the dominant factor in the disparity between the response times of the “SQL calls in a loop” approach and the stored procedure technique.

# Configuration

Figure 1 shows the configuration used for this study. The database manager, DB2 for z/OS V8, resided on a z990 server with 1 CP (single CPU) equipped with 8 GB of central memory that ran z/OS 1.6. Remote access to this server was achieved using an OSA (Open Systems Adapter) card supporting a 1 Gb/sec. data transfer rate.

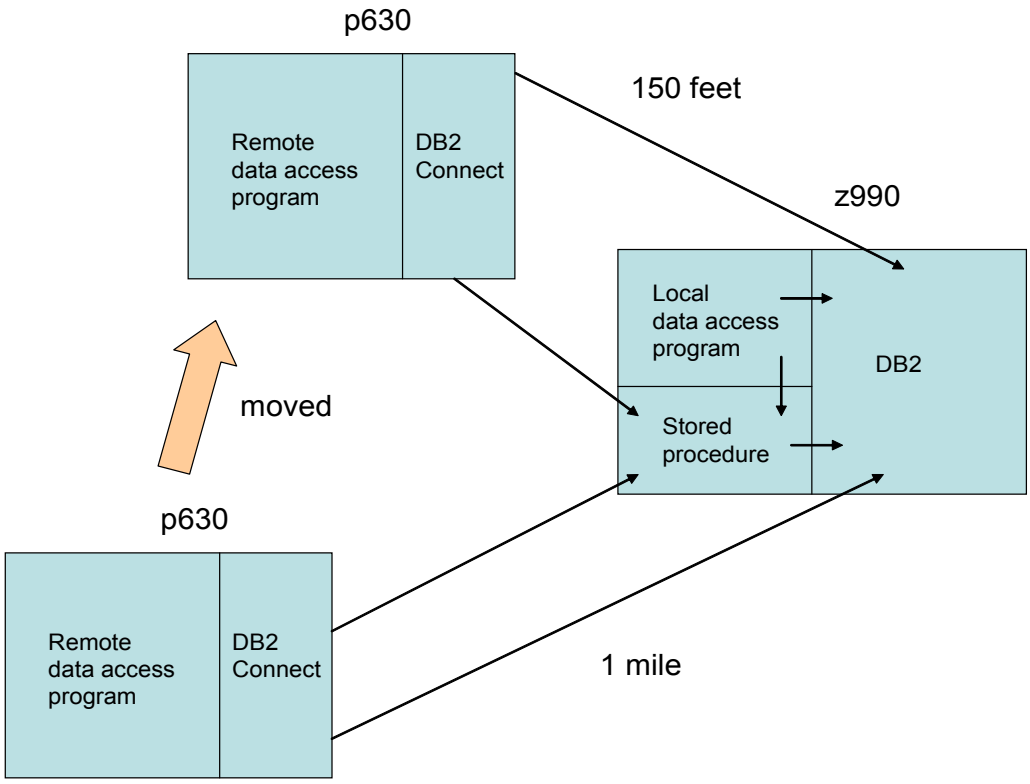


*Figure 1 Configuration*

For the test cases involving local data access, the test program ran on the same system as the database server. In this environment only the z990 server was used and the program connected to DB2 through the RRSAF (Resource Recovery Services Attachment Facility) interface.

For remote data access, a p630 server with four processors was used to execute the program. The server came with 8 GB of memory. AIX® 5.3 was the operating system and DB2 Connect™ V8.2 provided the connectivity to DB2 for z/OS on the z990 server. The test program communicated to DB2 Connect, which in turn sent the SQL requests to DB2. We installed the p630 server in two different locations. For the far-distance test cases, the p630 server was located in a building about one mile (network distance) away from the database server. Then we moved it to approximately 150 feet from the z990 server for the near-distance tests. The p630 server stayed on the same LAN before and after the move. In fact, the server's IP address, netmask, and default gateway did not change with the move. The objective was to maintain a similar network topology so that the possibility of introducing performance differences was minimized.

The stored procedure ran locally to DB2. For the local access test cases, the test program called the stored procedure, which then accessed DB2 data. For the remote access test cases, the test program utilized DB2 Connect to communicate to the stored procedure. As in the local data access situation, the stored procedure ran in the z990 server and accessed data directly.

## Local versus remote accesses

The workload consisted of a C program issuing a SELECT SQL statement one million times in a loop. No table data was accessed since the SELECT was performed against the SYSDUMMY table. When the program ran in local access mode, it used the RRSAF interface to communicate to DB2. Result set was passed to the application via memory. For the remote access test case, the C program ran in an AIX server. Since there were one million calls to DB2, there were one million round trips across the network.

Figure 2 on page 5 shows that it took 2 minutes 27 seconds to access DB2 data locally. Remote access was substantially longer, coming in at 10 minutes and 15 seconds. Network latency was the major contributing factor in this discrepancy of elapsed times. CLI traces were taken to estimate network latency, and results indicate a duration of approximately 500 microseconds for each round trip in this test configuration. This confirmed that network latency was the largest component in the test program execution response time (500 microseconds x 1,000,000 = 8 minutes 20 seconds, and 89 of the 500 microseconds was spent in DB2 processing).

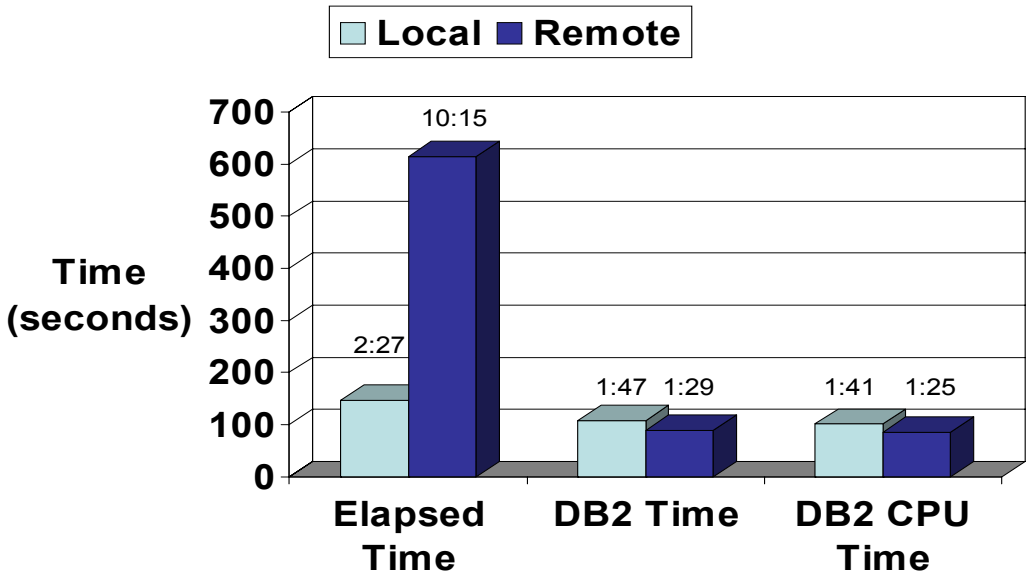


*Figure 2 Local versus remote access times*

We also collected DB2 accounting data during the measurements. DB2PM reports were then produced to provide DB2 elapsed times and DB2 CPU times. The DB2 elapsed time metric measures the time DB2 executes a SQL statement. Remote access mode was shorter because part of the processing was performed at the AIX server. By the same token, less CPU time was consumed in DB2 in this mode.

This test case illustrates the performance advantage of co-locating an application on the same server as the database manager. This is particularly important for applications that heavily use database manager services. An example is a program that performs data loading across multiple platforms. Instead of using the load utility of a database product, programmers often choose to perform inserts instead. That way they do not have to maintain multiple sets of load utility control statements, one for each platform the program supports. Using an insert strategy, the program can perform the inserts to database servers located on many platforms. But they play a price in performance. With the insert strategy, it is not uncommon that data loading takes multiple days. Co-locating the application and database components could reduce the loading time significantly.

## Stored procedure

The C program was modified so that its core logic was replaced by a stored procedure. The program remained running on the AIX server. During its execution it executed a single call to the stored procedure running locally to DB2. Although it was possible to continue using CLI in the stored procedure, we decided to change the stored procedure to use static SQL statements to follow development practices. Not surprisingly, the stored procedure version took substantially less time to execute, as shown in Figure 3 on page 6.

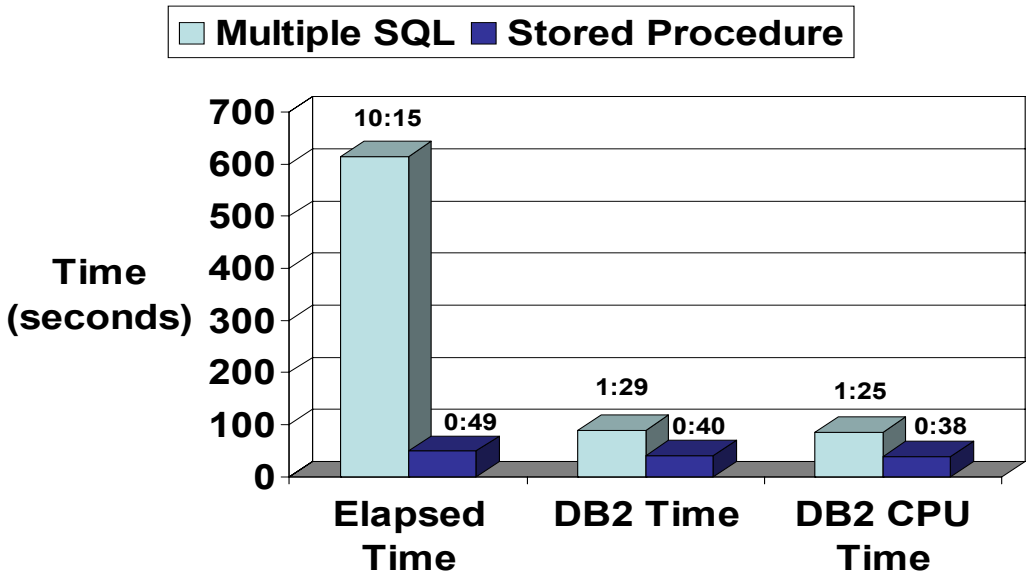


*Figure 3 Remote data access times*

Improvement in the test program execution time came from two areas: elimination of network latency and performance advantage of static over dynamic SQL statements. With just a single call to the stored procedure, the one million network round trips were eliminated. Moreover, static SQL statements consumed a smaller amount of CPU cycles than dynamic statements. These two factors explain the dramatic drop in response time from 10 minutes 15 seconds to 49 seconds.

As shown in Figure 4, improvement in DB2 time and DB2 CPU time came from using static SQL statements as well as elimination of the set up processing of one million remote SQL calls (no longer necessary to call Distributed Data Facility one million times to pass SQL statements to DB2). To the extent that the logic of a program lends itself to multiple SQL calls but returning a single result set, implementing the program with a stored procedure could lead to potentially significant performance improvements.

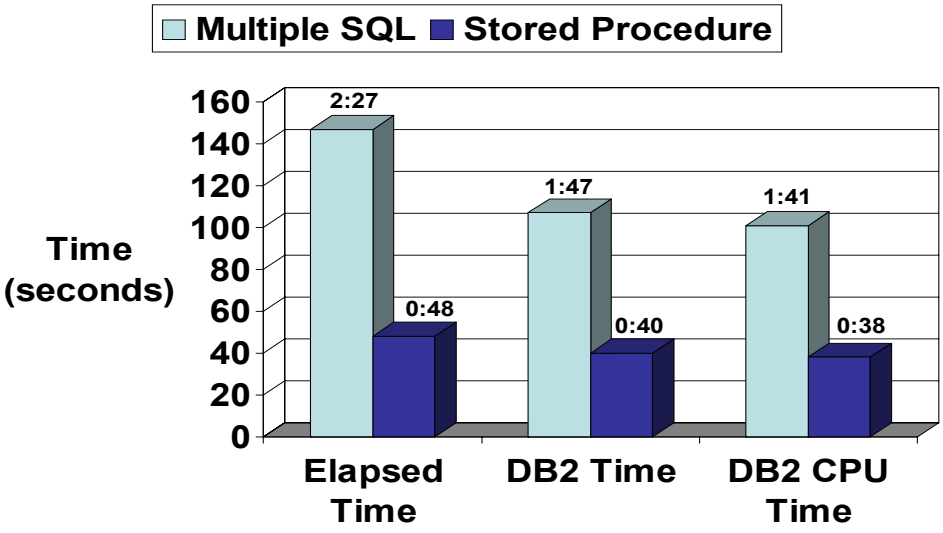


*Figure 4 Local data access times*

We ran another set of measurements to determine performance benefits of calling a stored procedure locally. In the base test case the C program ran in the server where the database manager was located. It issued SQL statements in a loop to access DB2 data. Then it called the same stored procedure used in the previous measurements. The stored procedure used static SQL statements to access DB2 data.

At first glance, data from Figure 4 on page 6 suggests that calling a local stored procedure outperformed calling DB2 in a loop by a large margin. But the performance benefits mainly come from using static SQL statements versus CLI. Simply moving the business logic of a program to a stored procedure in local access mode does not yield significant performance improvements. To verify this claim, the local program was modified to use static SQL statements to call DB2 in a loop. This was then compared with the original version of the program that utilized CLI (dynamic SQL). Figure 5 shows that static SQL outperforms dynamic SQL by a large margin.

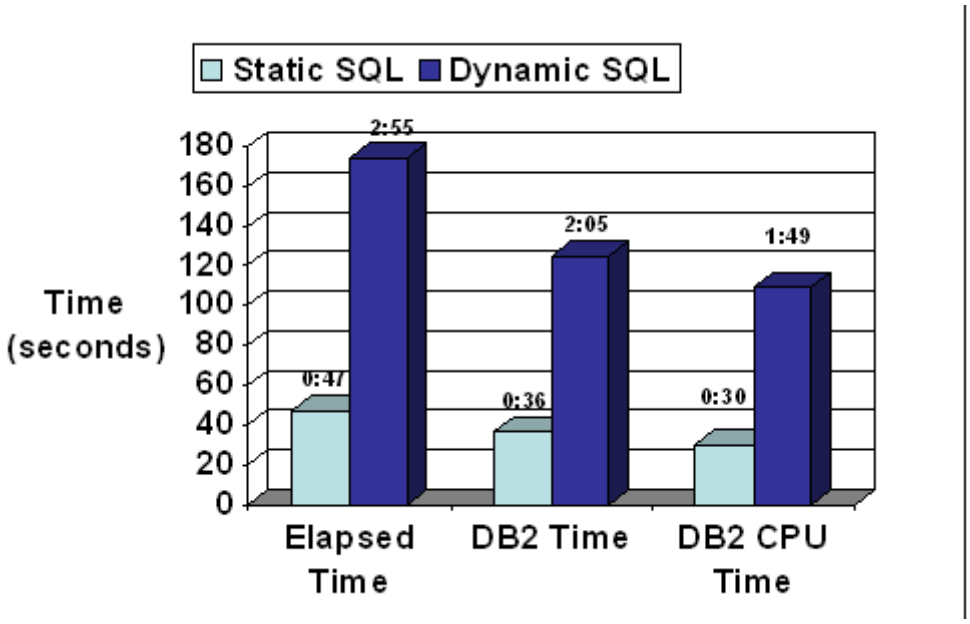


*Figure 5 Static versus Dynamic SQL Performance*

The difference in performance between static and dynamic SQL statements in production environments should be less than what is shown in figure 5. In this set of measurements the program used a single SQL statement to access the SYSDUMMY table and consumed minimal DB2 resources. For production applications, it is likely the SQL statements are more resource intensive, relegating the SQL preparation logic to a smaller percentage of the SQL processing. For the dynamic SQL measurement, statement caching was enabled with an achievement of 100% cache hit ratio.

## Remote access distance

The subject of network distance's impact to performance comes up very often in benchmark discussions. Will a database server located far away from the application server cause performance degradation? Is it absolutely necessary to house the database server next to the application server? To answer these questions, we conducted a series of measurements to determine performance impact based on the distance between the application and database servers. This was not meant to be a comprehensive study, and as such only two data points were captured in the study. The first configuration utilized an application server located 1 mile

away from the database server. Then the application server was moved to the same building as the database server, and the distance between the two servers was shortened to 150 feet.

As shown in Figure 6, performance was virtually identical between the two data points. Although it took slightly longer to run the test program for the “1 mile” configuration, the difference was insignificant. Also, one would not expect any difference in DB2 time and DB2 CPU time between the two configurations because these metrics are independent of network distance. Data from Figure 6 confirms this expectation.

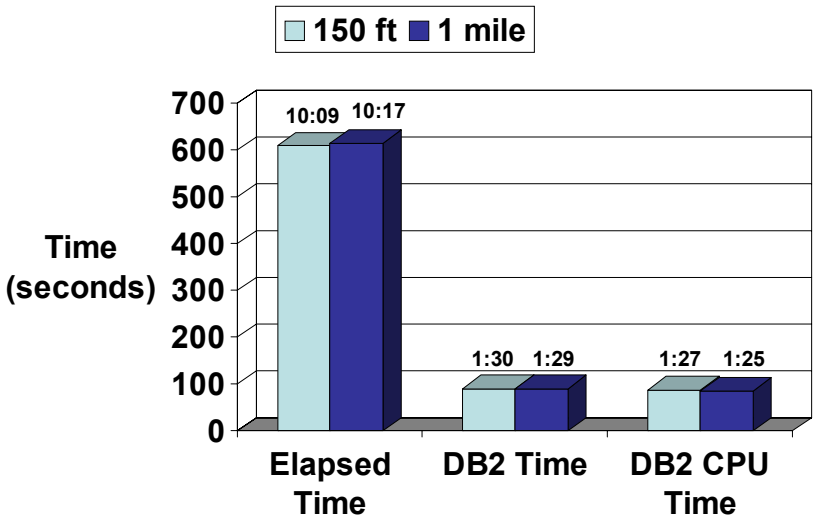


*Figure 6 Impact of distance to performance*

If a database server is located several miles away from an application server, network latency might affect application response time. We need additional measurements to confirm this suspicion. However, based on the measurements in this study, it is safe to assume that network distance of one mile or less does not affect response time noticeably. The results of this study are based on a configuration that resides entirely in a LAN.

**Note:** A configuration where the components are connected by a WAN probably yields different performance characteristics, which is a more likely scenario when the servers are located many miles apart.

# Summary

Applications that issue a substantial number of SQL calls, such as programs running in batch mode, can benefit from co-location with the database manager. Results from this study indicate elapsed time reduction of 4X by simply moving the test program to run on z/OS. In some cases, migrating a program from a distributed platform to a series server does not call for a significant amount of work. But for those situations where the migration effort is non-trivial, the cost can be minimized by using the Unix System Services (USS) in z/OS or the Linux® for zSeries® operating system. In the Linux option, performance improvement could still be observed if the Linux guest and the z/OS LPAR reside on the same physical server. The amount of improvement is reduced due to a network connection between z/OS and Linux; however, hipersockets deliver substantially higher speed data transfer than a real network.

Stored procedures could deliver a big performance boost for remote access applications. Instead of calling DB2 multiple times, therefore incurring the latency cost of multiple network

round trips, a program can call a stored procedure just once. Results from the tests indicate an elapsed time reduction of tenfold, although actual results vary from application to application.

We found in this study that network distance up to a mile does not affect data access time noticeably. This data is useful for IT professionals designing a production or a benchmark environment. With this information, a wider selection of configurations is deemed to be feasible to the designers.

# The team that wrote this Redpaper

This Redpaper was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Nin Lei** is an expert in DB2 performance and database design. One of his main interests is in Very Large Data Bases (VLDB), covering areas of database and application design and performance analysis. He was a frequent speaker at database and data warehouse conferences. He is currently an architect at the IBM® zSeries Benchmark Center conducting client studies.

## Additional authors

Leticia Cruz
Victor Chao

## Thanks to the following people for their contributions to this project

Mike Ebbers
International Technical Support Organization, Poughkeepsie Center

# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:
*IBM Director of Licensing, IBM Corporation, North Castle Drive Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law**: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE:
This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

This document created or updated on December 29, 2005.

Send us your comments in one of the following ways:

- Use the online **Contact us** review redbook form found at:
  **ibm.com**/redbooks
- Send your comments in an email to:
  redbook@us.ibm.com
- Mail your comments to:
  IBM Corporation, International Technical Support Organization
  Dept. HYJ  Mail Station P099
  2455 South Road
  Poughkeepsie, NY 12601-5400 U.S.A.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

| | | |
|---|---|---|
| @server® | zSeries® | IBM® |
| @server® | AIX® | Redbooks™ |
| Redbooks (logo) ™ | DB2 Connect™ | Linux™ |
| z/OS® | DB2® | |

The following terms are trademarks of other companies:

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.