

ActiveBase Performance

Performance improvement and control

Table of Contents

How large is the download and what does it include?.....	1
Do I need to change my Applications or Database?	1
What is the process of installing AB*Performance on Linux/Unix environments?	2
What are the resources AB*Performance consumes when it is installed on the database server?	2
How can we prevent AB*Performance from being a single point of failure?.....	3
Is it possible to have a unique SQL proxy (Active Base) for a set of applications/ databases or is it necessary to install a single Active Base instance for each application?	3
What are the system-requirements to install ActiveBase:	3
What is ActiveBase program language?	4
How do you upgrade AB*Performance?	4
What is maximum number of rules that can be implemented without effecting propagation?	4
How is ActiveBase different from native DBMS performance features and tools?	4
Can AB*Performance rewrite/add-hint or block SQL requests that will perform a full scan on large tables?.....	5
How AB*Performance knows what is the database execution of a statement – even before the statement is executed in the database?	5
How does AB*Performance manage Parallel Query degree?	5
How AB*Performance can improve performance for ad-hoc queries?. As each long query is identified through different patterns and tuned manually by the DBA's.	5

How large is the download and what does it include?

The download for the AB*Performance server software ~50MB, dependant on your OS version, and includes the installation as well as documentation and a management console installation.

How long will it take to install and configure AB*Performance on a single database?

Installation only takes about an hour. You will first install the server software (opening a tar), followed by a PC – administrator management consol.

Predefined rules will be available immediately, and you will be able to define custom rules

Do I need to change my Applications or Database?

No! AB*Performance is completely transparent to both applications and databases. NO DATABASE CONFIGURATION CHANGE IS REQUIRED!

The only configuration change required, is either:



1. Add a new entry to the Oracle client centralized routing flat file, (tnsnames.ora or oranames) with ActiveBase server host and listener port number. This file is centrally managed for all clients, thus a single and simple configuration change immediately affects all clients to be routed via the ActiveBase server. For testing purposes, a tnsnames.ora copy can be saved on a local directory on your client.

OR...

2. Change the database listener port to another port (hidden) and configure ActiveBase listener port to listen to the primary port (e.g. Oracle port 1521). ActiveBase includes specific functionality to support this type of configuration by enabling selective bypass to specific clients (clients are identified by an include/exclude list of program/host name and OSusers). This bypass enables all clients to connect to ActiveBase listener port, where specific clients will be routed with ActiveBase policies applied, and other applications will be routed directly to the hidden database listener, thus bypassing ActiveBase policies (e.g. ETL processes that do not require auditing or security policies).

What is the process of installing AB*Performance on Linux/Unix environments?

You create user active, and open AB*Performance tar file. It also comes with its OWN JVM installation, inside the installation tar. You 'start' AB*Performance by running ./start command from the installation directory. Edit the setPath file for specifying your specific installation directory.

After installation, you define the Oracle databases that run on the server. When introducing a database in AB*Performance, you need to enter oracle username and password. This Oracle user needs to have a plan_table and 'become user' privileges to the main schema.

What are the resources AB*Performance consumes when it is installed on the database server?

Overhead is negligible - about 1% CPU load, where propagation delay is about 150 microsecond per SQL statement. Keep note that AB*Performance switch capability enables to define the specific modules, clients and IP's that will go through the rule processing (and overhead – 'use-rules' routing action), and which will automatically bypass AB*Performance rules and connect directly to the Oracle listener ('direct' routing action).

A rule tree has been included, that enables ActiveBase to bypass all fast and efficient SQL statements with no delay, while only the long and inefficient SQL statement are caught and manipulated by the rules – accelerating them by x10-1000 times.

How can we prevent AB*Performance from being a single point of failure?

Several high-availability features ensure consistent service:

1. Install AB*Performance as a cluster – either on two servers with an IP takeover between them or even two installations on the same server, where each is configured to listen on a different listener port.
2. Oracle itself guarantees ActiveBase high availability: Oracle clients have a built in failover mechanism (both Oracle client failover and Oracle TAF – Transparent Application Failover). When an Oracle listener fails, Oracle automatically connects to a secondary listener. AB*Performance configuration takes use of the Oracle failover mechanism. If ActiveBase listener is down, the application automatically reconnects DIRECTLY to the Oracle listener. These mechanisms guarantees that in extreme circumstances all clients will automatically and immediately reconnect directly to the database server, completely bypassing AB*Performance server.
3. AB*Performance has also a “bypass all routing” option, that when activated, redirects all clients directly to the databases, bypassing ActiveBase.

Is it possible to have a unique SQL proxy (Active Base) for a set of applications/databases or is it necessary to install a single Active Base instance for each application?

You can have a single ActiveBase server support multiple applications/databases. For example, a common configuration is to have AB installed on the database server in production, and another single AB server for supporting multiple applications in QA/test/preproduction environments.

What are the system-requirements to install ActiveBase:

Oracle versions (client and / or database) include all clients/application/databases running on Oracle8.0 until 11g

Type and OS version – Windows, Linux and Unix (AIX 5.2 or higher, Windows2000, Solaris 7 or higher, HP-UX 11 or higher, Linux RedHat 7.1)

Windows Service pack level is not necessary

All computer processor are supported except Windows Itanium 2

Minimal hard drive space is 800 mega for full auditing

Network and system ports include 8185, 4000 - 4010, 1526 on the server we install ActiveBase

No other components (system and or software) need to be installed before.

Installation user:

On windows, install using an administrator account.

On Unix/Linux, a user active needs to be created with an active directory.

All installation is done inside this active directory.

Minimal screen resolution - at least 600*800

What is ActiveBase program language?

Java 1.5.0_04, which is included within ActiveBase installation, so no special preparation is required prior installation.

How do you upgrade AB*Performance?

Simply stop the server process (using ./stop command within the activeknowledge directory), download the tar file into active directory and untar it. Upgrading does not delete your existing installation configurations. You also will need to uninstall ((control panel -> install/uninstall programs->ActiveBase),) and reinstall the Windows client to be able to connect and administer the new version.

What is maximum number of rules that can be implemented without effecting propagation?

No real limit or affect. AB*Performance testing is done with up to 10000 rules.

Nevertheless based on our experience large installations do not have more than 30 - 50 active rules (although each rule fixes many different SQL statements – using ‘search and replace’ rewrite and regular expression tag place holders, partial text matching, partial execution plan matching etc.- resulting in a small BUT VERY EFFECTIVE number of rules in large production implementations. Note that in massive on-line-transaction-processing (OLTP) application, we apply rules only on selected modules as well as use only regular expression matchers and rewrite actions for best speed.

How is ActiveBase different from native DBMS performance features and tools?

Native DBMS performance tuning tools typically provide SQL optimization suggestions, with no capability of implementing this suggestion (requires manually changing the application source code, when applicable). This is not feasible when the customer has no control over the application source code (in packaged OLTP applications), and in large DW and BI environments - where reports and ad-hoc queries are created and changed daily. Oracle's DBMS Outline feature does not support applications using dynamic SQL requests and are impossible to administer and manage on a large scale. ActiveBase broad rule capabilities can fix these occurrences immediately and with no changes whatsoever.

Why is AB*Performance blocking action better and safer then blocking provided by other solutions?

Only AB*Performance enables to block a specific SQL requests in any application (2, 3 or n-

tier applications) without touching application code, while returning a customized notification to the user (multi-language supported).

Application connections are not torn and sessions are not killed. Only the specific request is blocked. Other requests from different users using the same connection continue with no application abstraction whatsoever.

Can AB*Performance rewrite/add-hint or block SQL requests that will perform a full scan on large tables?

Yes, AB*Performance can identify the execution plan of the SQL statement, the cost and the from clause objects and apply rewrites, hints and statement blocks accordingly.

How AB*Performance knows what is the database execution of a statement – even before the statement is executed in the database?

AB*Performance checks the cost and execution plan of the SQL statement by using a JDBC connection from ActiveBase server to the database, and executing 'get explain plan for SQL'. It parses the cost and the execution plan steps returned by the database and tests these results with the execution plan and cost based matching rules.

Note: if no rule includes execution plan or cost identification criteria, NO 'GET EXECUTION PLAN' IS DONE.

How does AB*Performance manage Parallel Query degree?

AB*Performance can add parallel hint that reduces or increases the parallel degree of a request. In addition, it can disable parallel execution all together by adding 'alter session disable parallel' command on incoming sessions.

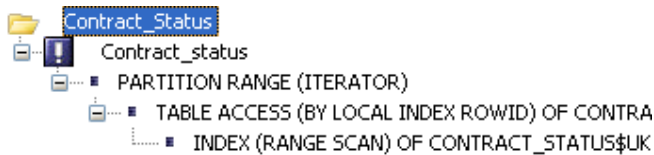
How AB*Performance can improve performance for ad-hoc queries?. As each long query is identified through different patterns and tuned manually by the DBA's.

AB*Performance includes several mechanism that have been specifically designed to improve ad-hoc queries:

1. More powerful: as ActiveBase performance rules identify and fix reports based on PARTIAL explain plan matching, PARTIAL SQL text identification, number of partitioned scanned – rules identify performance problems and fix ALL AD-HOC QUERIES WITH SIMILAR PERFORMANCE PROBLEMS!, where a single ActiveBase rule using partial explain plan pattern identification fixes tens and hundreds of reports and ad-hoc daily with similar performance problem.

Naturally there will always be other ad-hoc queries that will not be identified and fixed by ActiveBase rules, yet the powerful rules puts in place a clear and efficient methodology that makes the current process a much more efficient one.

For example, the following identification:



Is applying a hint `/*+NO_INDEX*/` to all reports and ad-hoc queries that include within their execution plan the following three actions: Partition range, Table access and index range scan INSTEAD OF PARTITION RANGE SCAN – extremely inefficient as there is a relevant partition key in this table. The rule automatically adds the hint `/*+NO_INDEX*/` which causes Oracle to use the partition correctly. As the customer experienced many reports and ad-hoc queries with this problem, the rule is extremely efficient, running minutes instead of hours.

2. More efficient: ActiveBase central rule management enables to manage and reuse all tuning effort done by the DBA team. No time is wasted at tuning ad-hoc queries that have been already tuned in the past.
3. Faster: ActiveBase provides an SQL Expert utility that saves DBA time by automatically tuning the SQL request

www.active-base.com

400-00101-302 | 02/09 | © 2009 ActiveBase, Ltd. All rights reserved. All other third-party trademarks are the property of their respective owners.

