
insure/INDEX

User Guide and Reference

Version 8.0

Document date: 09/15/2011

Centerfield Technology, Inc.

<http://www.centerfieldtechnology.com>
© 1999-2011 Centerfield Technology, Inc.

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the written permission of Centerfield Technology, Inc.

Notice: The information contained in this document is subject to change without notice. Centerfield Technology, Inc. shall not be liable for errors contained herein or consequential damages in connection with the furnishing, performance, or use of this material.

The sample source and resulting programs are provided for your use only. You may repackage any source or resulting program or portion of source or resulting program provided within this document for the purpose of product distribution, installation, or system archiving. Any other use requires the written permission of Centerfield Technology, Inc.

Centerfield Technology, Inc. and HomeRun are trademarks of Centerfield Technology, Inc.

All other product and brand names are trademarks and registered trademarks of their respective companies.

Use of this book and the information within implies an understanding and agreement with the above statements.

Centerfield Technology, Inc.
3131 Superior Drive NW - Suite C
Rochester, MN 55901

1	OVERVIEW OF HOMERUN	5
1.1	PRODUCT CONTENTS	6
1.1.1	<i>HomeRun installation</i>	6
1.1.2	<i>User Guide and Reference</i>	6
1.2	PRODUCT LIBRARIES	6
2	INTRODUCTION TO INSURE/INDEX.....	7
3	GETTING STARTED WITH INSURE/INDEX.....	8
3.1	HOME RUN REQUIREMENTS AND INSTALLATION.....	9
3.1.1	<i>Product Requirements</i>	9
3.1.2	<i>Product Installation</i>	9
3.1.3	<i>Connecting to the server for the first time</i>	18
3.2	SECURING THE USE OF HOME RUN TOOLS.....	21
3.3	USAGE TRACKING AND DATABASE PROFILES	23
3.3.1	<i>Understanding Database Profiles</i>	23
3.3.2	<i>Creating a database profile</i>	25
3.3.3	<i>Controlling the Usage Tracker's use of resources</i>	38
3.3.4	<i>Importing SQL activity information</i>	39
3.4	IDENTIFYING IMPORTANT INDEXES WITH INSURE/INDEX.....	41
3.5	MANAGING CHANGES TO YOUR DATABASE	42
3.6	CREATING INDEXES RECOMMENDED BY INSURE/INDEX	44
4	AUTONOMIC DATABASE ASSISTANT (AUTODBA).....	47
4.1	OVERVIEW	47
4.2	GETTING STARTED.....	47
4.3	ADVICE	49
4.4	MISCELLANEOUS RIGHT-CLICK MENU OPTIONS.....	52
4.5	CONFIGURATION	53
4.5.1	<i>Thresholds</i>	53
4.5.2	<i>Permissions</i>	55
4.5.3	<i>Intervals</i>	56
4.5.4	<i>Miscellaneous settings</i>	56
4.5.5	<i>Filters</i>	57
4.5.6	<i>Schedule</i>	58
4.6	EXECUTE NOW	60
4.7	LOGS	62
4.8	PRIVILEGES	62
4.9	OPTIONS	63
5	SUPPORT.....	64
5.1	CONTACT INFORMATION	64
5.2	ADDITIONAL INFORMATION	64
6	BUILT-IN TOOLS.....	65
6.1	USING VISUAL SQL EXPLAIN	65
6.1.1	<i>Visual SQL Explain plan constructs</i>	65
6.2	USING DATABASE EXPLORER	77
7	APPENDIXES	78

7.1	INTRODUCTION TO DATABASE PERFORMANCE MANAGEMENT.....	78
7.1.1	<i>Effective performance management.....</i>	78
7.1.2	<i>Understanding an SQL statement's implementation</i>	80
7.1.3	<i>Managing change in a production environment.....</i>	81
7.1.4	<i>Understanding side effects.....</i>	82
7.2	SYSTEM-WIDE INSTALLATION IMPACTS.....	83
7.2.1	<i>System Values</i>	83
7.2.2	<i>Servers, Jobs, and Monitors</i>	84
7.2.3	<i>Database Monitor and HomeRun</i>	84
7.2.4	<i>Subsystems and Work Management.....</i>	84
7.2.5	<i>Exit points.....</i>	87
7.2.6	<i>TCP/IP Usage.....</i>	87
7.2.7	<i>Job Scheduler Usage</i>	88
7.2.8	<i>Default public authority of libraries.....</i>	88
7.2.9	<i>Command Language (CL) Commands Used by HomeRun.....</i>	88
7.2.10	<i>OS/400 System APIs Used by HomeRun.....</i>	89
7.2.11	<i>Programs Adopting *OWNER Authority</i>	90
7.3	HOME RUN TUNING METHODOLOGY FAQ.....	93
7.3.1	<i>Why tune your database and queries?</i>	93
7.3.2	<i>What types of data access tuning can you do?</i>	93
7.3.3	<i>How do you know what to tune?</i>	94
7.3.4	<i>What types of things can I do to improve my data access performance?</i>	94
7.3.5	<i>Specifically, how does HomeRun help me?</i>	95

1 Overview of HomeRun

The HomeRun toolset allows you to efficiently design, deploy, manage, and support all aspects of an SQL-based environment. The set of tools in HomeRun includes:

- insure/INDEX
- insure/ANALYSIS
- insure/MONITOR
- insure/RESOURCES
- insure/SECURITY

All tools make use of the HomeRun server on the iSeries. The password you enter on the server controls which tools are available for your use. However, the manual for each of these tools is included in your product download.

In addition, the following tools are included with every HomeRun installation:

- Autonomic DBA
- Visual SQL Explain (a.k.a. sql/OPTIMIZER)
- Database Explorer
- Lock Detector
- Semaphore Wait Detector
- Mutex Wait Detector

See the Appendixes for more information on using these built-in tools.

1.1 Product Contents

The product you received from Centerfield Technology, Inc. contains the following items:

1.1.1 HomeRun installation

The download contains the software for both your iSeries and your Windows 2000/XP/Vista/7 client workstation. Refer to the Product Installation section for instructions on installing the software.

Also, for JDE OneWorld® (aka Oracle E1®) environments there is an additional piece of software that needs to be installed on the Windows® Terminal Server (i.e. Citrix® server).

1.1.2 User Guide and Reference

Manuals for all tools in the HomeRun toolset can be found in your product download. Each manual is intended to help you get started with the tool, explain the tool's features, and provide guidance on the effective use of the product.

1.2 Product libraries

Centerfield uses a naming convention where all IBM System i™ object names are prefixed with the letters “XC”.

As part of that naming convention throughout this manual the Centerfield *{program library}* will be “XCENTER80” and the Centerfield *{data library}* will be “XCENTERD80”.

This version of the product uses TCP/IP port number 9920 by default. The rest of this document refers to this port as the default port “**{default port}**“. If that port is in use, the installation will incrementally search for unused port numbers starting at **{default port}** and continue looking until one is found. To determine the port that was chosen use the following command:

```
WRKSRVTBLE SERVICE(CENTERFIELD_SERVER_80)
```

2 Introduction to insure/INDEX

insure/INDEX allows you to collect database performance information and identify indexes that are key to affecting SQL performance in your environment. insure/INDEX provides options that allow you to focus your analysis on specific libraries and files. It also provides tracking mechanisms that record index recommendations and advice that has been implemented on your system.

The insure/INDEX tool consists of two functions:

- Usage tracking
- Index manager

The Autonomic Database Assistant (AutoDBA) also provides index management functions that include identifying unused, redundant, and suboptimal indexes. For further information on AutoDBA, see the “Autonomic Database Assistant” section.

3 Getting started with insure/INDEX

This section describes how to get started quickly with insure/INDEX. With the information in this section, you should be able to create the new indexes that will be most beneficial to your users.

The steps outlined in this Getting Started section are:

1. Install the product
2. Secure the use of insure/INDEX for authorized users only
3. Define a database profile within insure/INDEX to record the database activity that is occurring in your environment.
4. Use the insure/INDEX wizard to identify the key indexes needed on your system for better performance.
5. Establish the policies you are going to use to manage the changes you make to your database.
6. Create the recommended indexes.

3.1 HomeRun requirements and installation

3.1.1 Product Requirements

The HomeRun toolset requires specific System i and Windows hardware and software before it will install and perform correctly.

Here is the list of required IBM PTFs:

- ✓ V5R4:
 - MF39418, MF39419, MF39466, SI24100, SI23714, SI23713, SI24580, SI23891, SI23890, SI24569, SI24504, SI23485, SI25668, SI25041, SI23365, SI23514, SI28951, SI28952, SI28953, SI30076, SI30013, SI30014, SI31631, SI31633
- ✓ V5R4M5:
 - SI24100, SI23714, SI23713, SI24580, SI23891, SI23890, SI24569, SI24504, SI23485, SI25668, SI25041, SI23365, SI23514, SI28951, SI28952, SI28953, SI30076, SI30013, SI30014, SI31631, SI31633
- ✓ V6R1:
 - SI31727, SI31641, SI31407

NOTE: PTFs listed above are current with the release date of this document. Centerfield may learn of new PTFs after the document release so we strongly encourage you to check our website for latest PTF requirements – www.centerfieldtechnology.com

3.1.1.1 System i

- ✓ i5/OS V5R4, or V6R1, V7R1
- ✓ TCP/IP configured

3.1.1.2 Windows

- ✓ Windows 2000, Windows XP, Windows Vista, Windows 7
- ✓ Minimum 80486 @ 66Mhz, 32 MB RAM
- ✓ 50 MB of disk space for product
- ✓ TCP/IP installed and configured
- ✓ ODBC driver installed

3.1.1.3 Citrix Windows Terminal Server for E1

- ✓ Windows NT Server, Windows 2000 Server, Windows 2003 Server

3.1.2 Product Installation

The HomeRun toolset has software that needs to be installed on the System i and on the workstations. The installation processes for both the System i and the workstations are

designed to be simple interfaces that provide good default values. The following sections in combination with the on-screen documentation help you understand the installation processes.

If you're re-installing HomeRun you **MUST** refer to the "**HomeRun Installation Guide**" document rather than these 1st time installation instructions.

3.1.2.1 System i Installation

- 1) Before beginning the installation on the System i, please review this list of the impacts the installation will have on your system. For more detailed information about the HomeRun System i installation, see the Appendix titled *System-Wide Installation Impacts*.

- ✓ This product uses TCP/IP to communicate between the System i and the personal computer. TCP/IP needs to be configured before the HomeRun server will function. Before continuing, make sure your Windows PC client can *ping* the System i system. See one of the following IBM System i references for information on configuring TCP/IP.
 - *TCP/IP Fastpath Setup (SC41-3430)*
 - *TCP/IP Configuration and Reference (SC41-3420)*
- ✓ The installation uses TCP/IP port number **{default port}**. If that port is in use, the installation will search for unused port numbers starting at **{default port}** and continue looking until one is found. To determine the port that was chosen use the following command:

WRKSRVTBLE SERVICE(CENTERFIELD_SERVER_80)

- ✓ The HomeRun server installs into a library named **{PROGRAM LIBRARY}**. *The installation process will copy over existing programs if the library exists.*
- ✓ The HomeRun data files install into a library named **{DATA LIBRARY}**. *The installation process will selectively replace the data files and preserve information specific to your configuration and settings if they exist.*
- ✓ The HomeRun server requires an active subsystem for proper installation and operation. Determine the subsystem you wish to use. It will be needed later in the installation process. If you have no special work management requirements, you may use the default of **{PROGRAM LIBRARY}/XCSBS80**.

-
- ✓ If you are installing over the top of an existing HomeRun or Database Essentials product library, the jobs currently using that library will be ended before the installation begins. The server will be re-started at the end of the installation if a valid password is entered during the install, or if you have installed over an existing library that already has a valid password.
- 2) Sign-on to a 5250 session with QSECOFR or a user profile that has security officer special authorities. This user profile must have a minimum of *ALLOBJ, *IOSYSCFG, *JOBCTL, *SECADM, and *SAVSYS special authorities.
 - 3) The system value QUSEADPAUT must be set to *NONE.

You can verify and change this system value using the following command.

WRKSYSVAL QUSEADPAUT

In addition, if the system value QUSEADPAUT is secured with an authorization list, the user profile used for the installation must be on that authorization list.

- 4) Place the CD-ROM containing the HomeRun product into your System i primary CD-ROM drive and issue the following command (does not apply when performing a fully electronic installation).

LODRUN *OPT

- 5) When you are prompted to configure the HomeRun server subsystem, either take the default (i.e. XCSBS80) or enter the subsystem where you want the HomeRun server and client jobs to run, and press Enter. It is recommended that the default be taken. You can reconfigure the server's subsystem after the installation by running the CFGSVR command.
- 6) When you are prompted to enter the HomeRun license password, enter the license password that was provided with your product and press Enter. The license password is based on the serial number of the System i and the LPAR number. Hence, a different license password is needed for each System i that you install.

If you do not know the license password for your System i, you can skip this step of the installation by pressing F12. You can enter the license password after the installation by running the PASSWORD command from the *{PROGRAM LIBRARY}* library.

If you defer adding the license password, the HomeRun server will not start automatically. No client machines will be able to connect to the System i using any of the HomeRun tools until the password is entered.

-
- 7) You will be returned to an System i command line when the installation is complete. If you did not skip any of the installation steps, the HomeRun server will be automatically started for you. If you skipped any steps in the installation process, you should perform the configuration steps that you skipped before you continue.
 - 8) If you skipped configuring the server or the license password as part of the installation, or if the server did not automatically start as part of the product installation, you need to start the HomeRun server before clients can connect to the System i using any of the HomeRun tools. Prior to trying to start the server, you should ensure that TCP/IP is active on your system. If it is not active, issue the following command using the correct options for starting the TCP/IP servers on your system.

 STRTCP STRSVR(*YES/*NO)

- 9) Issue the following command to start the HomeRun server:

{PROGRAM LIBRARY}/STRSVR

- 10) Verify the server is active in the subsystem that you specified. The job name will be XCSERVER. The following command can be used to check for the active job and to verify that the subsystem is correct.

 WRKACTJOB JOB(XCSERVER*)

- 11) The server will need to be restarted every time the associated subsystem is ended or the system is restarted. You may add the *{PROGRAM LIBRARY}/STRSVR* command to your system startup routine (**recommended**) or add an autostart job entry in the subsystem if you would like to automatically start the server. If you start the server in your system startup procedure, the i5/OS command STRTCP must be issued and TCP must be completely started before the server will start.
- 12) If you plan to use insure/SECURITY or insure/MONITOR, you will need to register Centerfield's exit point programs with the System i registration facility:

{PROGRAM LIBRARY}/ADDMON

This enables the policies you define to take effect. More information can be found in the manuals for each of these tools.

NOTE: you will have to recycle the host servers for the exit point registration to take effect (refer to the table on next page as well as the Appendix for more detailed information).

-
- 13) If you plan to use insure/RESOURCES, you must install policy managers before your resource policies will take effect. This can be done either with the:

{PROGRAM LIBRARY}/ADDQC EXIT

or with the menus in insure/RESOURCES PC client GUI.

ODBC exit may need recycling database host server and corresponding prestart jobs (i.e. ENDHOSTSVR *DATABASE, ENDPJ QUSRWRK QZDASOINIT, STRHOSTSVR *DATABASE)

More information can be found in the insure/RESOURCES manual.

- 14) You are now ready to begin the workstation installation.

3.1.2.2 Windows 2000/XP/Vista/7 Installation

We recommend Administrator authority on the PC to run the install:

- 1) Go to website: <http://www.centerfieldtechnology.com/downloads.asp>
- 2) Click on the download link for HomeRun
- 3) Follow the online instructions in the START HERE pdf.
- 4) If the installation needed to replace DLLs, you will be prompted to restart your system.
- 5) You may now use the HomeRun tools for which you have a license.

3.1.2.3 Installation instructions for OneWorld® customers using Windows® Terminal Server (i.e. Citrix® server)

These instruction apply **only** to customers that are running JDE OneWorld® (a.k.a. PeopleSoft EnterpriseOne® aka Oracle E1®) on their System i and have purchased insure/Monitor for OneWorld® license. Furthermore, customers using Terminal Servers (TS) in their environment and require Centerfield windows service running on those TS to achieve correlation of System i ODBC jobs to the OneWorld users will follow these instructions.

If these conditions do **NOT** apply to your scenario, do **NOT** follow these installation steps.

These instructions are to be performed by the Terminal Server administrator (i.e. OneWorld® CNC specialist) with Administrator authority to the server.

- 1) Locate **Centerfield.exe**, **Centerfield.dll** and **CenterfieldServiceConfiguration.exe** on the installation CD using Windows Explorer
- 2) Create a new folder on the Terminal Server (i.e. C:\Centerfield)
- 3) Copy and paste **Centerfield.exe**, **Centerfield.dll** and **CenterfieldServiceConfiguration.exe** to the newly created folder
- 4)

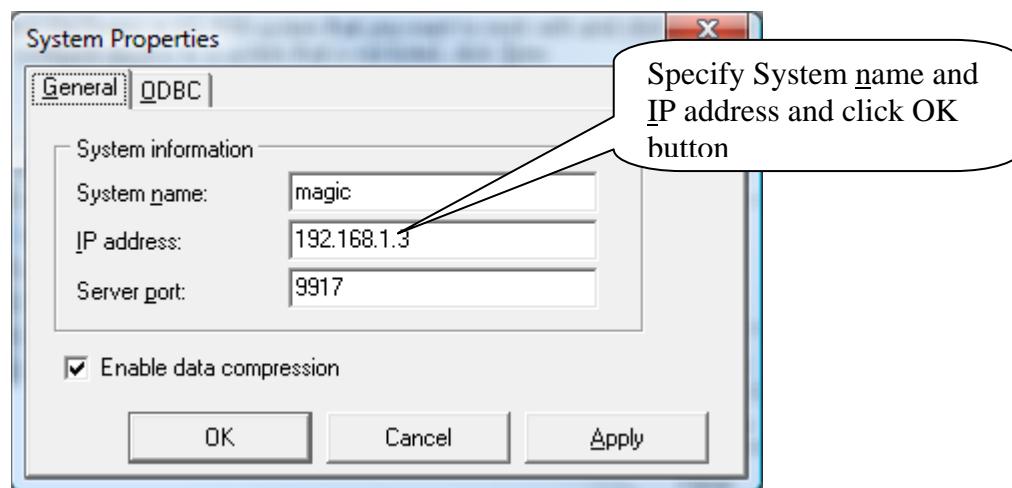
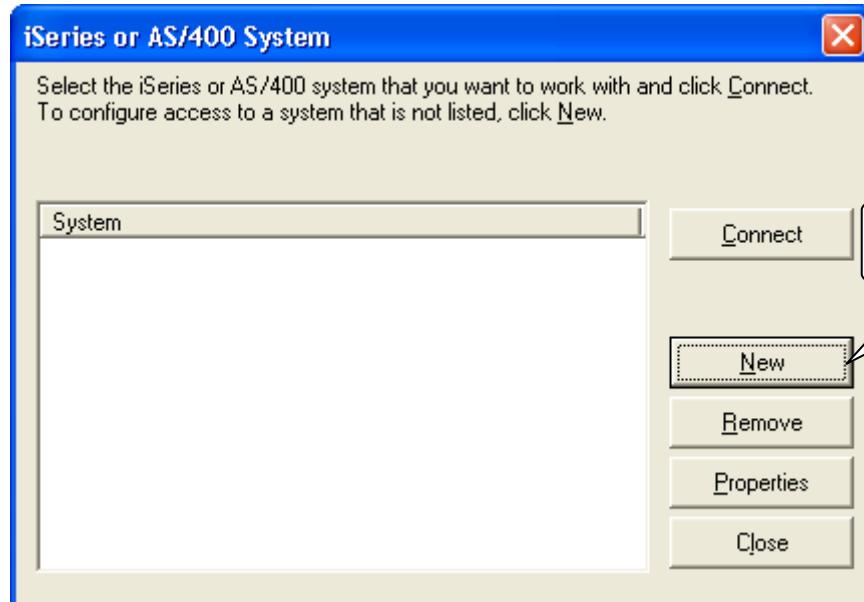
The screenshot shows a Windows Command Prompt window with a blue title bar and a black body. The title bar displays 'C:\WINDOWS\System32\cmd.exe'. The command line shows the following sequence of commands:

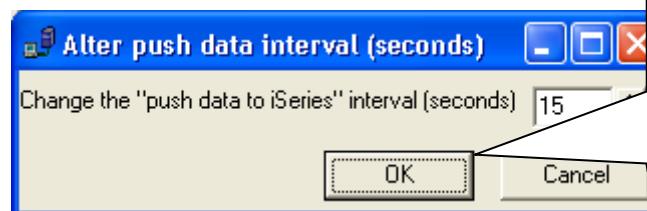
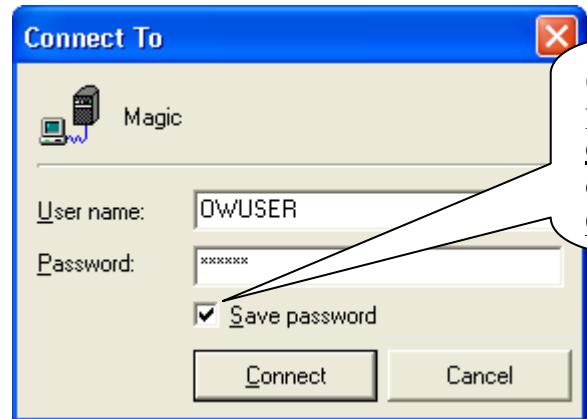
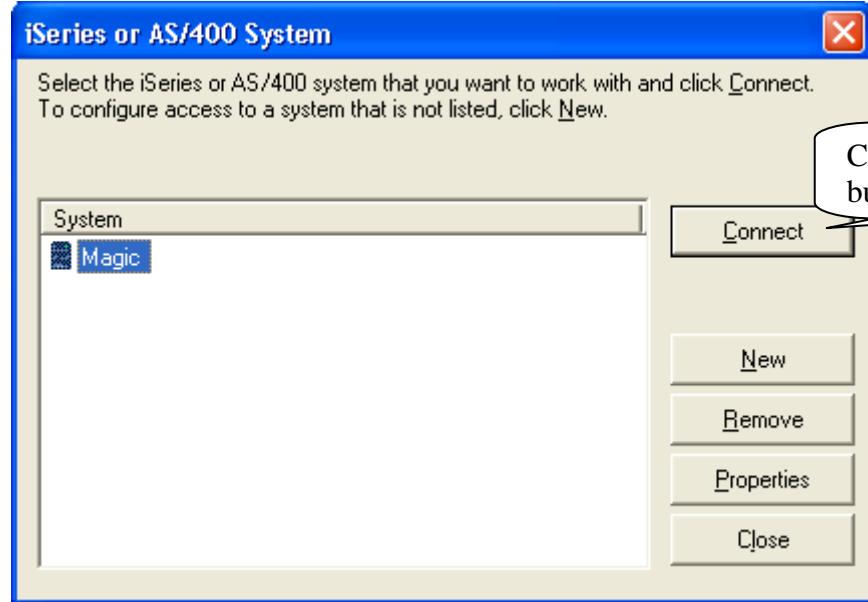
```
C:\Documents and Settings>cd C:\Centerfield  
C:\Centerfield>Centerfield /install
```

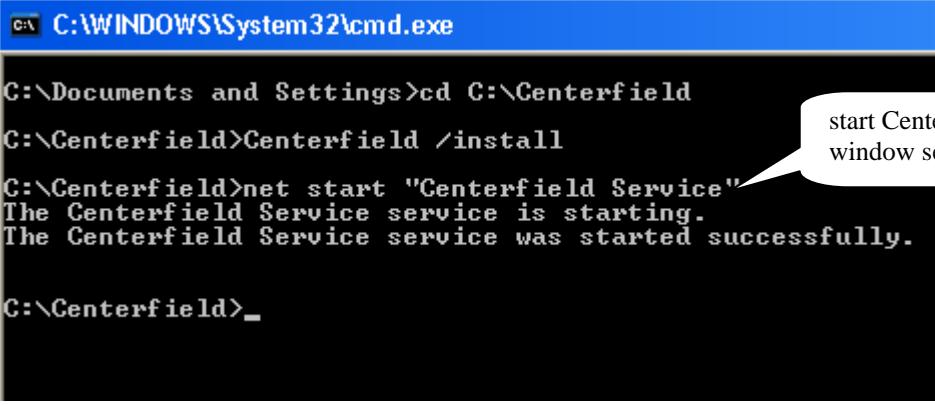
Three callout boxes with arrows point from the numbers in the list above to specific parts of the command line:

- 1 - Start DOS command prompt on the Terminal Server (points to the first 'cmd' in the command line)
- 2 - Set current directory to where the Centerfield files are located (points to the 'cd' command)
- 3 - Invoke service installation wizard (points to the '/install' option)

-
- 5) Several one time configuration windows will appear.





6) 
C:\WINDOWS\System32\cmd.exe
C:\Documents and Settings>cd C:\Centerfield
C:\Centerfield>Centerfield /install
C:\Centerfield>net start "Centerfield Service"
The Centerfield Service service is starting.
The Centerfield Service service was started successfully.
C:\Centerfield>_

start Centerfield window service

You can check Centerfield service status in a couple of ways. One is to invoke the services applet via Start->Run->services.msc. Another is to check for the existence of Centerfield.exe process in the Task Manager on the Terminal Server.

If there are active OneWorld connections on the Terminal Server, there should be an XCCLIENT job on the System i with the joblog message:

“Serving Terminal Server at IP address 0.0.0.0.” where 0.0.0.0 will list the true IP address of the Terminal Server. If there are no active OneWorld connections on the Terminal Server, Centerfield service will not push any data to the System i.

This completes the Centerfield service installation on the Terminal Server. Service should auto-start every time Terminal Server reboots. For the Centerfield service to push data, XCSERVER job needs to be active on the System i. You can start it by executing *{PROGRAM LIBRARY}/STRSVR* command, or automate this by adding STRSVR command to your IPL startup routine (DSPSYSVAL QSTRUUPGM).

There is one final configuration step that has to be performed from the insure/MONITOR for OneWorld® PC client.

Refer to the documentation located under section **“Configuration for OneWorld® customers”** in the **“insure Monitor User Guide and Reference”** document for details.

3.1.3 Connecting to the server for the first time

Once you have the product installed, you are ready to connect to the server from the Windows based workstations. The workstation installation adds a program group to your start menu called *Centerfield HomeRun*. Within the program group there are one more features that can be selected depending on the options that were selected at installation time. To start the HomeRun tool you want to use, choose its name from this group.

The HomeRun tools use a consistent interface for connecting to the System i. When you start one of the HomeRun tools for the first time on your workstation, you will need to configure your connection to a server.

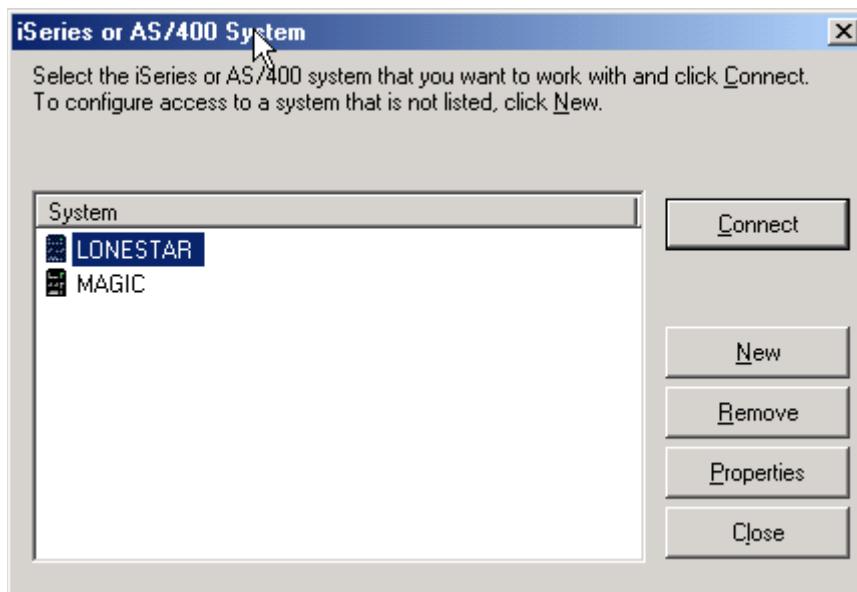
Auto-configuration

When you start the tool, it will try to auto-detect the System i systems in your environment and configure itself to connect to them.

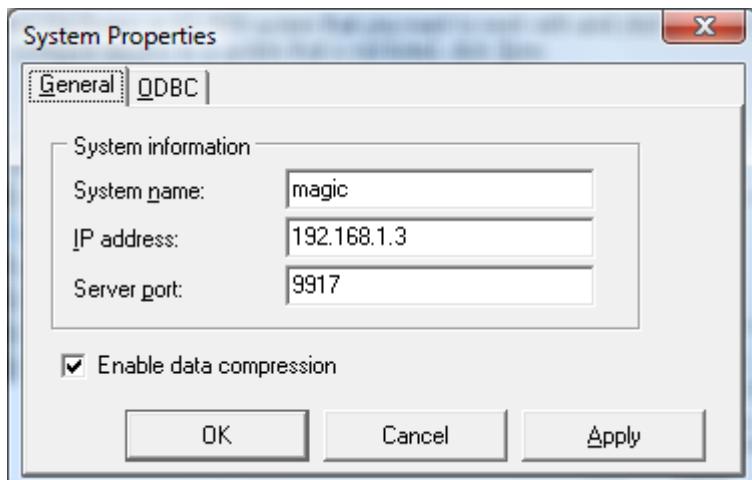
The auto-configuration interface will show you all of the systems that it can detect. If no systems are detected, the auto-configuration will be skipped and you will need to define your system connection information manually.

Manual configuration

If the auto-configuration interface did not detect your system, or you want to manually configure a connection, use the following instructions.



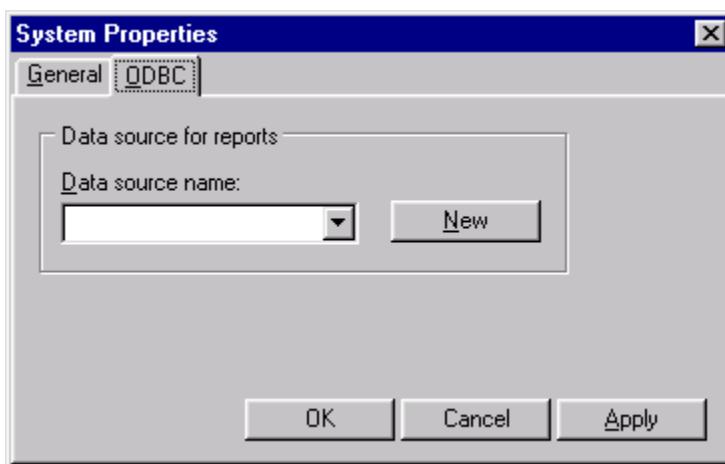
When the *System i system* window is displayed, press [New] to add a new system.



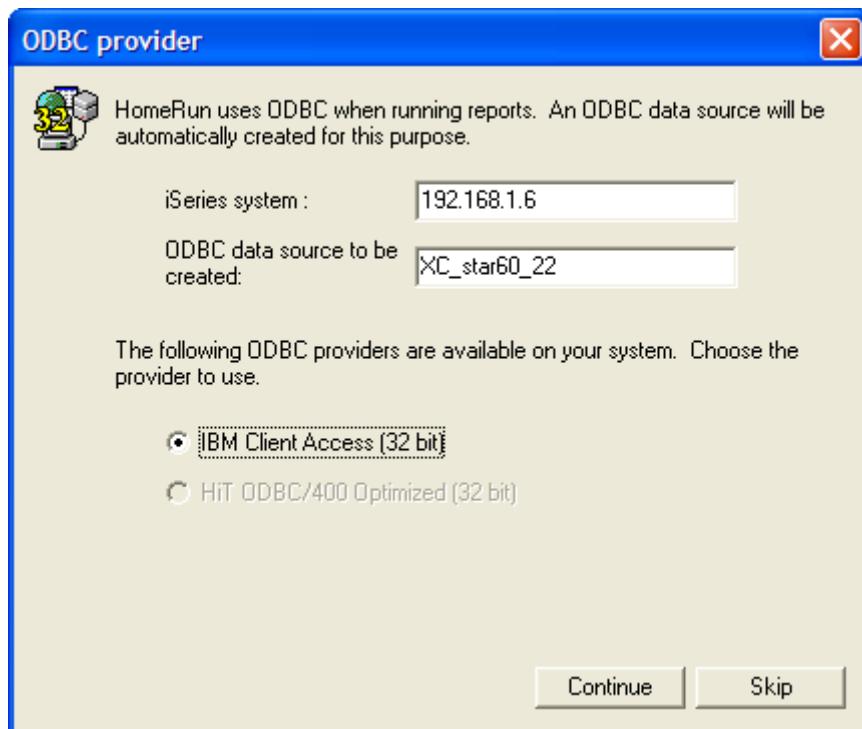
For *System name*, enter a descriptive name that identifies your system.

For *IP address*, enter either the symbolic name or dotted IP address for your System i system.

For the *TCP/IP port*, use the port that was selected during the System i installation process, by default {**default port**} is used.



On the *ODBC* tab, specify a name for your ODBC data source and press [**New**] to create a data source for this system. If you already have a Client Access ODBC data source that is configured to your system, you can specify your existing data source. If you choose to create a data source, you will see a display similar to the following.



Specify the name of the System i as defined in your connection software, and select the data source type that you want, and press [**Continue**]. You must have either Client Access or HIT ODBC installed to use this fast path interface for creating your data source. If you have another ODBC provider, you must create the data source with the ODBC Administrator utility that is provided with Windows and then specify the data source name within the *System Properties* window on the *ODBC* tab.

After you have created your data source and specified values for all of the appropriate prompts on the *System properties* window, press [**OK**] to create the connection definition.

Connecting to a system

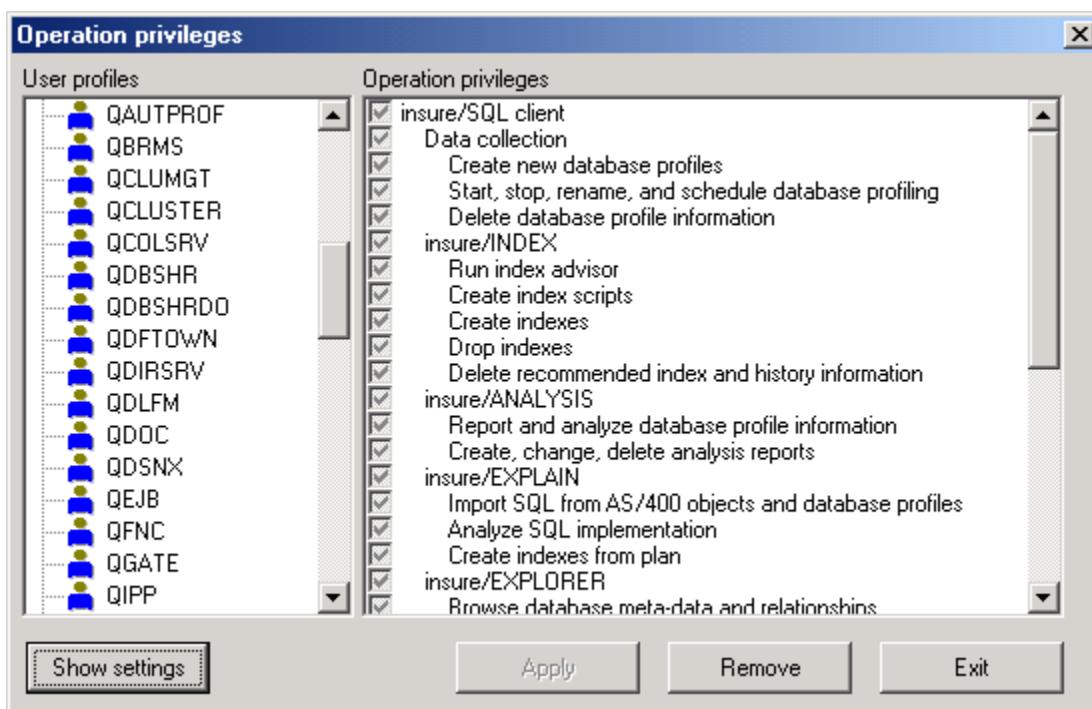
Select the system that you want to work with from the list provided in the *System i system* window, and press the [**Connect**] button. Enter your user profile and password and press [**Connect**].

3.2 Securing the use of HomeRun tools

HomeRun provides a broad suite of powerful functionality to the IT organization. Often, duties are split among various specialists. In this situation, it may be desirable to control what functions in the toolset each specialist can use. HomeRun provides a set of granular controls to allow an IT administrator to determine who can and should use each part of the toolset.

To implement security settings for a user or group profile use the following procedure:

1. In any of the HomeRun tools, select Tools | Privileges.
2. The next dialog has a list of users and groups on the left side and an itemized list of product features on the right as seen below:



3. To quickly restrict usage to a few users, select the *PUBLIC user profile and unclick the top-most box. At the same time, you should make sure one user profile has the capability to update privileges. To do that, pick one user profile and make sure *Administration* and *Edit HomeRun privileges* are checked. If you do not do this step, only the QSECOFR profile will still be able to edit privileges.
4. To change the settings for a particular user, click on their profile and push the "Show settings" button. On the right you will then see the parts of the product that profile is

authorized to use. If the box is gray, the user does not have explicit authority defined and will get authority from either their group profile or global settings (the authority set for *PUBLIC).

5. To restrict a user from a particular feature, simply uncheck the box in front of that feature and click the “Apply” button. If the user attempts to use that function, they will be given a message saying they are not authorized to that feature and that they should contact their system administrator.
6. To remove settings, choose a profile (or set of profiles with multi-select) and click “Remove settings”. This will result in all settings being picked up from the group profile or *PUBLIC.

NOTE: When settings are changed for a user, they will not take effect until the user connects to that iSeries again (i.e. exits the client GUI and restarts it).

3.3 Usage Tracking and Database Profiles

The first step in optimizing your database or identifying a performance problem is to record the activity that is occurring. HomeRun refers to this activity as **database profiling**. Both insure/INDEX and insure/ANALYSIS have a Usage Tracking feature which records and collects database performance information into a **database profile**, also called a **collection profile**.

3.3.1 Understanding Database Profiles

A database profile has several important attributes including name, type, and collection time periods. These correspond to the tabs you will initially see in the Profile Definition dialog box.

3.3.1.1 Profile Name

When you create a new database profile, you are asked to give it a name. This name should uniquely identify the profile and thus should not be a duplicate of an existing profile. You will use this name later when you work with the database profile in insure/INDEX or insure/ANALYSIS. It is suggested you name your profiles to include:

- The purpose for the data collection.
- The date it was created.
- Who created the profile.

Since all users of the HomeRun toolset will be able to see all profiles, this naming convention will help everyone using the product identify the purpose and owner of each profile.

3.3.1.2 Collection Type

The collection type dictates which jobs on your system are included in the profile.

The most exhaustive method is to collect for **all jobs** on your system. The advantage of this approach is that you can quickly get a complete picture of all query activity happening against your database. The disadvantage is that the collection may require a significant amount of disk storage and additional CPU resource. If your environment does not have a high rate of SQL-based insert, update, and delete transactions (for example, you are collecting information for an IBM System i™ used as a data warehouse), then collecting data for all jobs is a good place to start. If you are resource-constrained, then it is recommended you use other methods described next.

Another approach is to collect information for a subset of **currently active jobs**. The advantage of this approach is that you can pinpoint specific jobs or users to collect data

for and thus do targeted analysis. The disadvantage of this approach is that once a database profile is started you cannot add new jobs to that profile.

The third approach is to pre-define **criteria to select jobs** of interest. With this method you can specify a job name, a job user name, or a subsystem. For example, you may want to monitor all jobs running queries using IBM's ODBC driver. To do this you would specify the job name of the ODBC servers (e.g. QZDASOINIT if they are connected with TCP/IP). The advantage of this method is that you can target certain jobs rather than collecting data for the whole system. Another advantage is that new jobs that come into the system that meet your criteria will automatically become part of the active profile.

If you specify more than one type of criteria, they will be combined to identify the specific jobs of interest. For example, you may want to monitor all jobs that start with QP and jobs in subsystem QS SERVER. To do this, simply specify QP* for the job name to identify the generic job name on the "Job Name" tab and specify QS SERVER on the "Subsystems" tab.

If more than one item is listed in the criteria (for example you want jobs in the QS SERVER or QBASE subsystems), add both names to the list. Multiple choices for a single category (like "Job Name") are "ORed" together. Criteria for multiple categories are "ANDed" together. For example, if you want to monitor all jobs with a job user of JOE in subsystems QINTER or QS SERVER you can simple enter these criteria.

3.3.1.3 Collection Time

The final attribute of a database profile is the time period chosen to collect data.

One method is to dynamically start and stop profiles. You might use this approach when doing interactive analysis or trying to diagnose a performance problem. If you start the profile it is important to remember that it must be stopped. If you fail to stop a profile, you may collect a large amount of data and thus consume a lot of disk space.

The second method is to schedule data collection for predefined periods of time. For example, you may know that many reports are run on Friday morning. If you wish to find out who runs them, against what files, and if there are performance problems, a profile can be set up to automatically collect information every Friday morning. The advantage of a scheduled profile is that you do not have to remember to start or end a profile since it is done automatically for you.

The Usage Tracker also lets you simply create the profile and not start or schedule it. To use the profile at a later time, it must either by started interactively or scheduled.

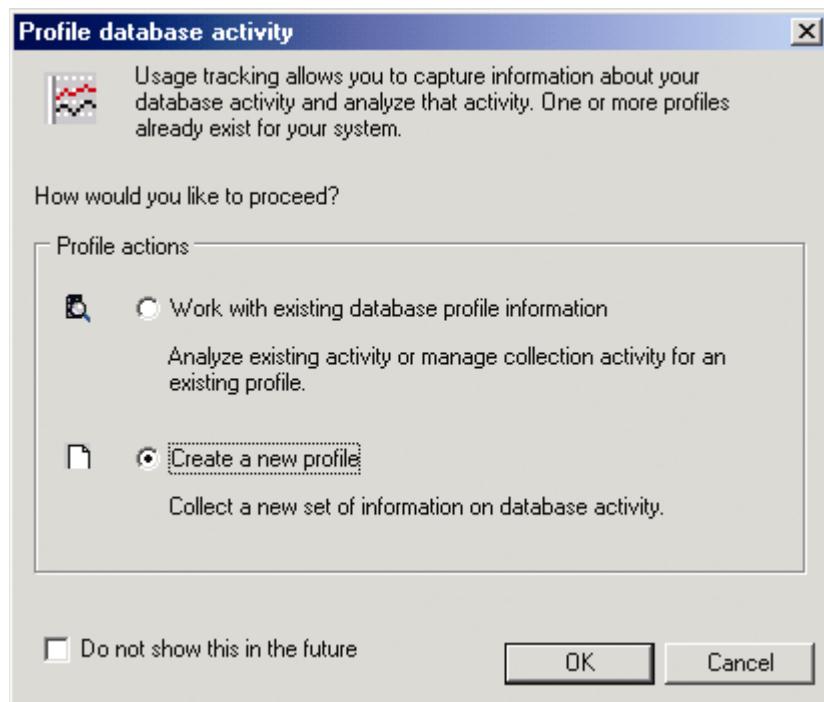
3.3.2 Creating a database profile

To create a new database profile:

Access the Usage Tracking window by doing one of the following:

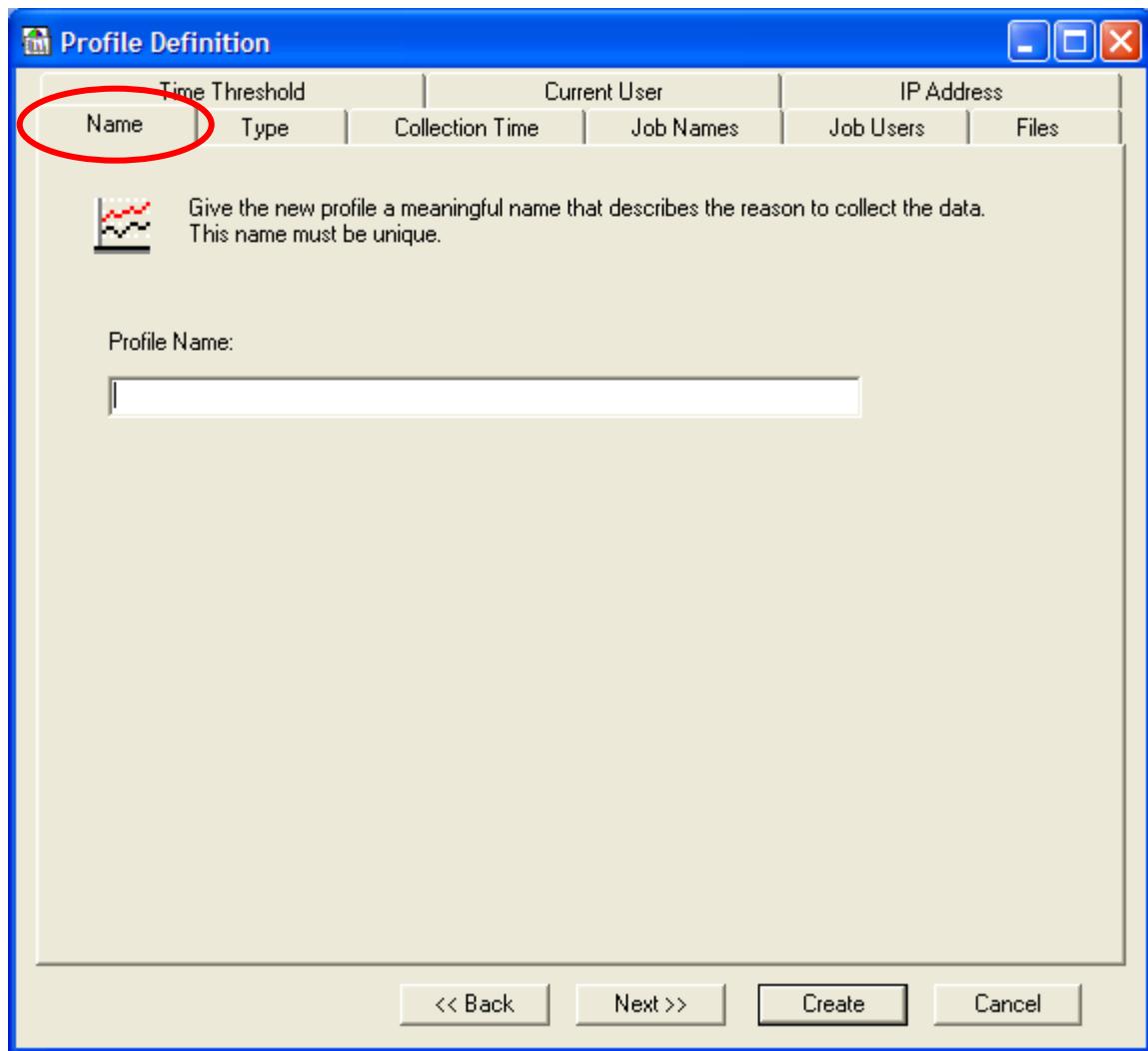
- From the *Welcome* screen, select *Track database activity*.
- From the *Tools* menu, choose *insure/INDEX* or *insure/ANALYSIS*, then choose *Usage Tracking*.

You will be presented with the following window. Choose to create a new profile.

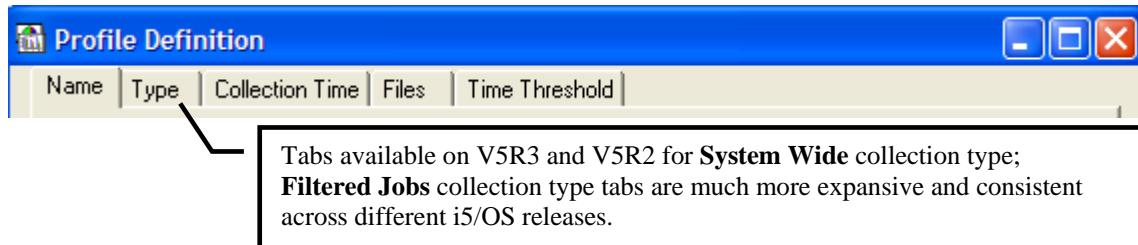


NOTE: some of the tabs in the following screen captures are release and/or PTF level dependent, so they may not appear on your specific installation.
All of the following screen captures were captured on i5/OS V5R4 current on PTFs.

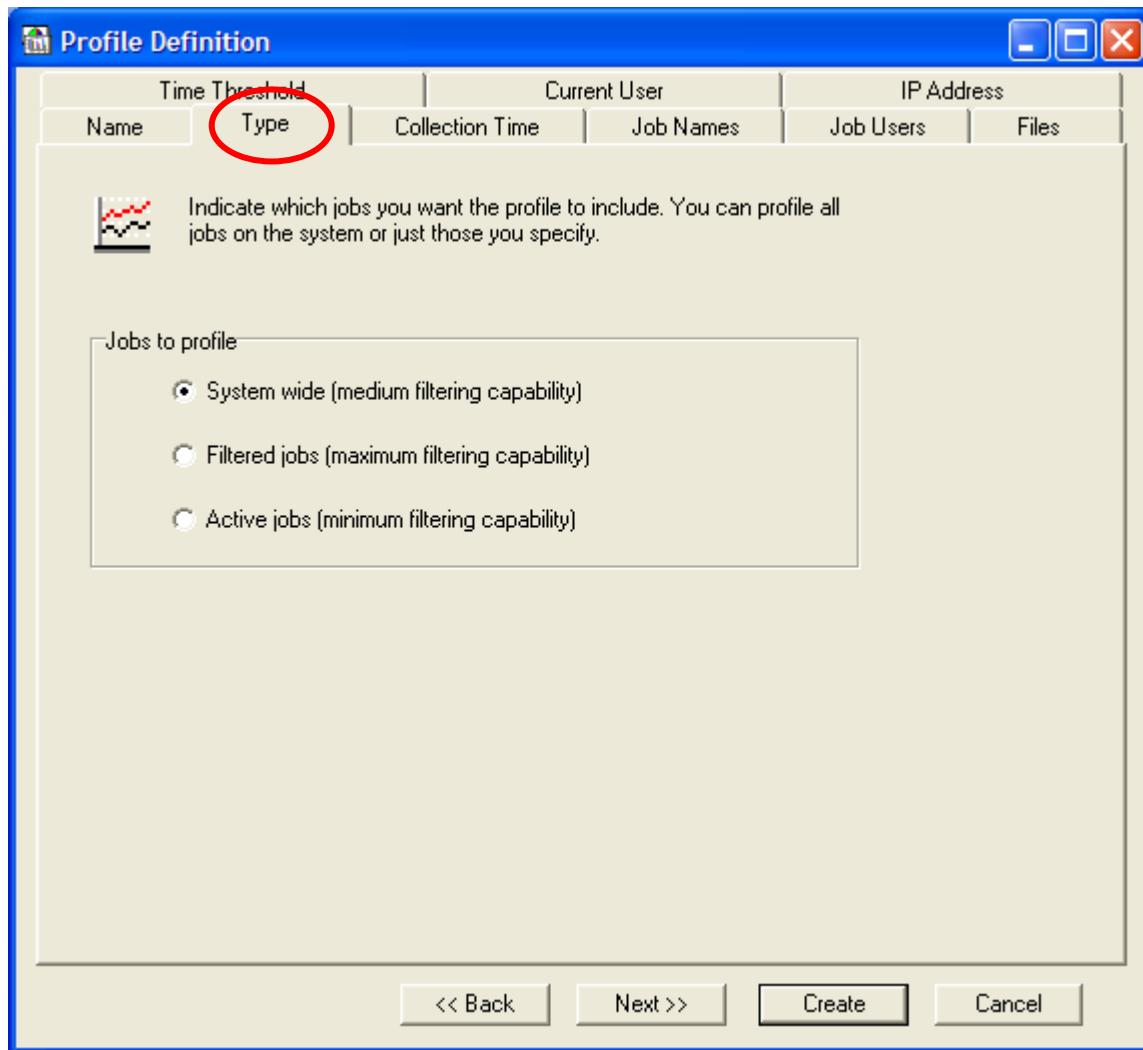
The *Profile Definition* window will pop up, with the *Name* tab showing. Enter a name that identifies the activity you are going to capture. Press the [Next] button to move to the next tab.



On V5R3 and earlier releases of the i5/OS you will subset of tabs for **System wide** collection type:



The next tab is the *Type* tab. If you choose a collection type other than *System wide*, you will see additional tabs appear that allow you to specify the additional criteria.



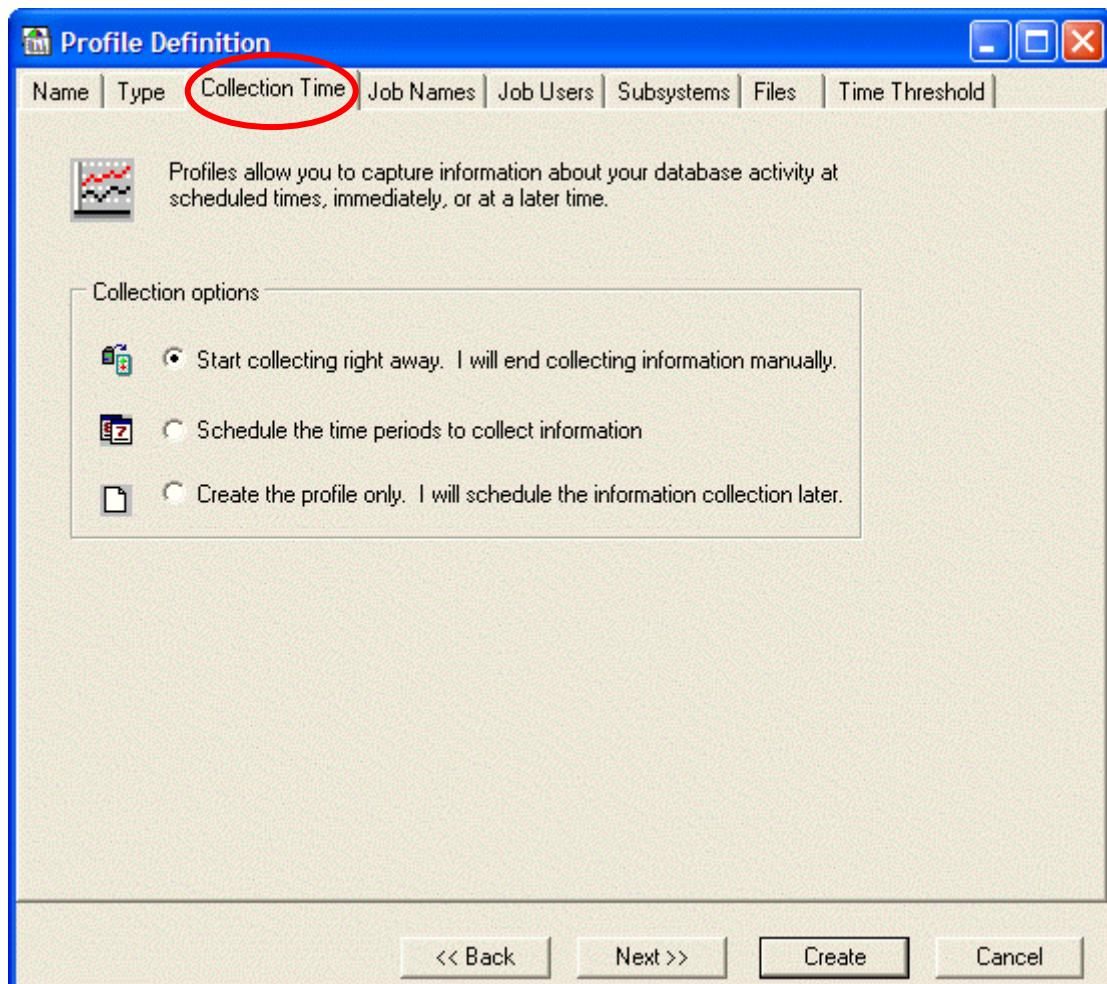
If you choose to collect *System wide* with little or no filters, you should be aware of these restrictions:

- The IBM System i™ STRDBMON command should not be actively collecting data (outside of HomeRun) when HomeRun attempts to start a collection to monitor all SQL activity.
- The IBM System i™ ENDDBMON command should not be used to stop collecting system-wide SQL activity while using this collection technique.

Starting with V5R4 version of i5/OS up to 10 public (system wide) filters are allowed.

When you have selected a Profile type, press [Next].

The collection time tab presents the following options:



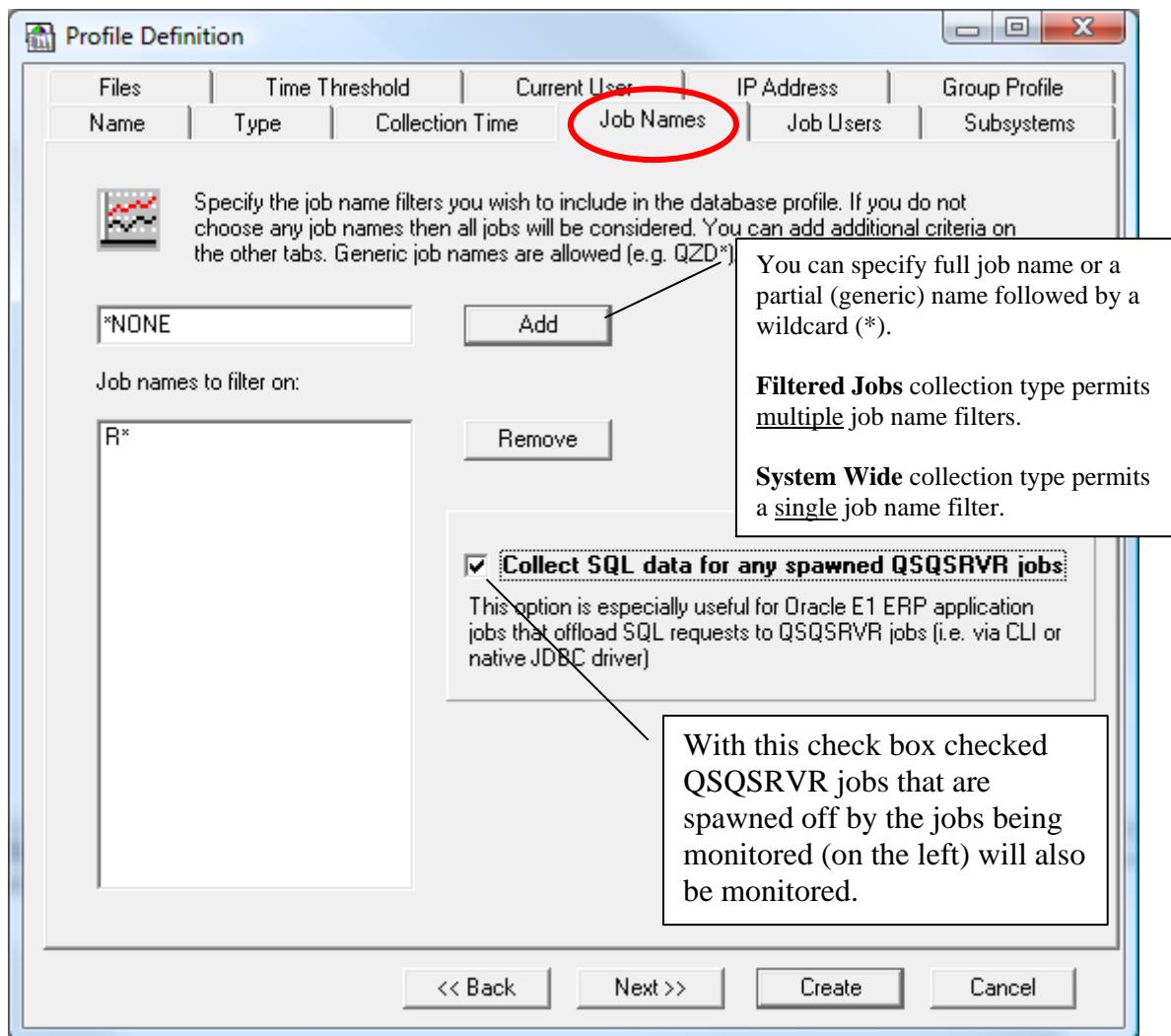
If you want to start collecting information right away, choose to ***Start collecting right away***. You will be responsible for stopping the collection after the activity you want to analyze is complete. This information is stored on the IBM System i™ in several files. If you forget to stop collecting information, these files will continue to grow and may use a lot of your IBM System i's™ storage. So make sure you do not forget to stop the collection if you start it manually.

You can also schedule the data collection. The Usage Tracker will automatically start and stop the collection at the times you specify. This way you do not have to remember to start and stop the data collection yourself. The schedule information is maintained on the IBM System i™, so you do not have to have the Windows PC active for the scheduling to work.

If you choose to only create the profile, a profile will be created without any collection time specifications. You will have to explicitly start it or schedule it at a later time.

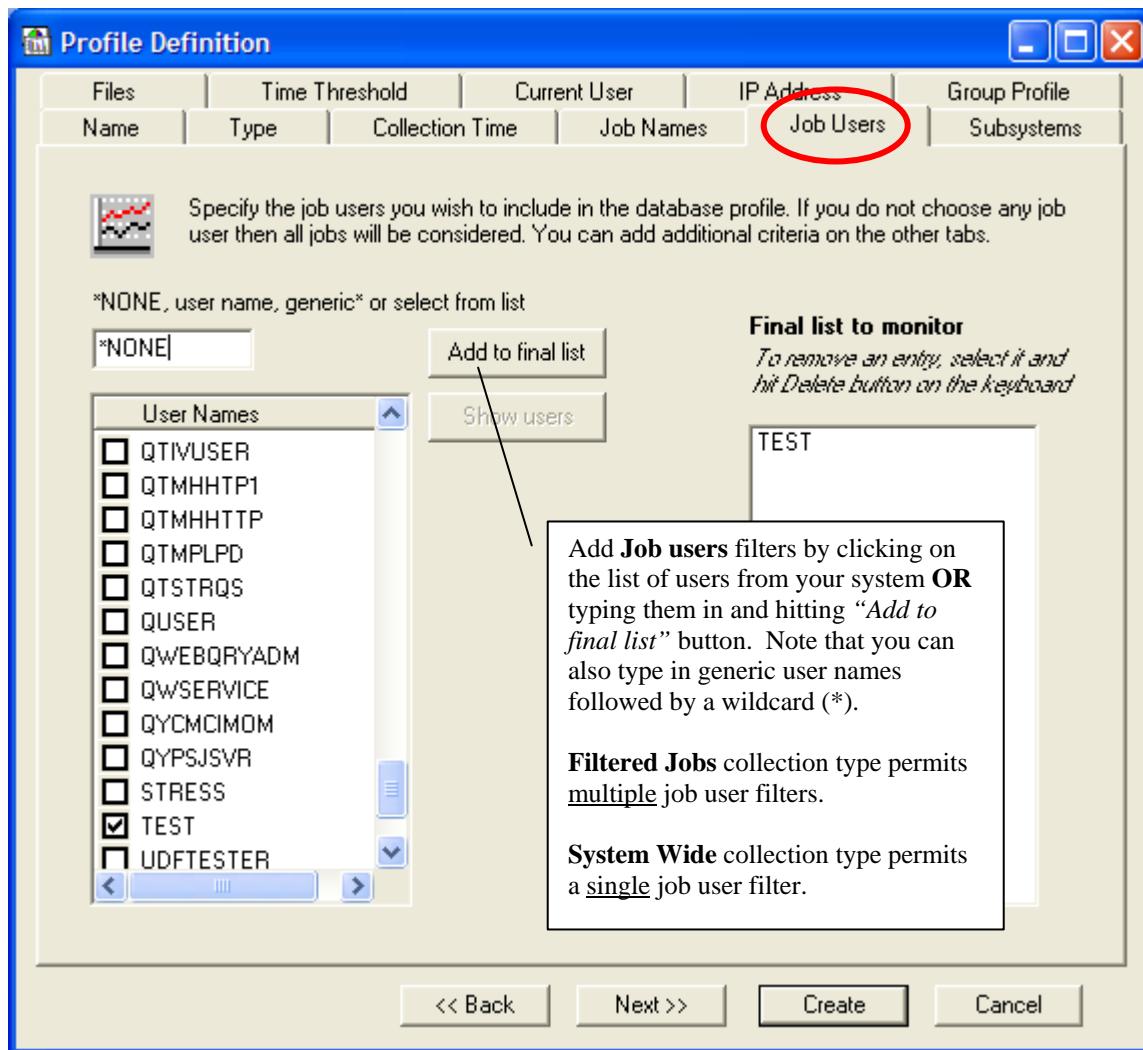
Once you have selected a collection time option, press **[Next]**.

The **Job names** tab allows you to choose the names of jobs that you wish to profile. If you have a common naming convention for jobs, you can specify generic names such as BCH* to include all jobs that start with the letters “BCH”. The following example will include all jobs that start with “QZDASOIN”.



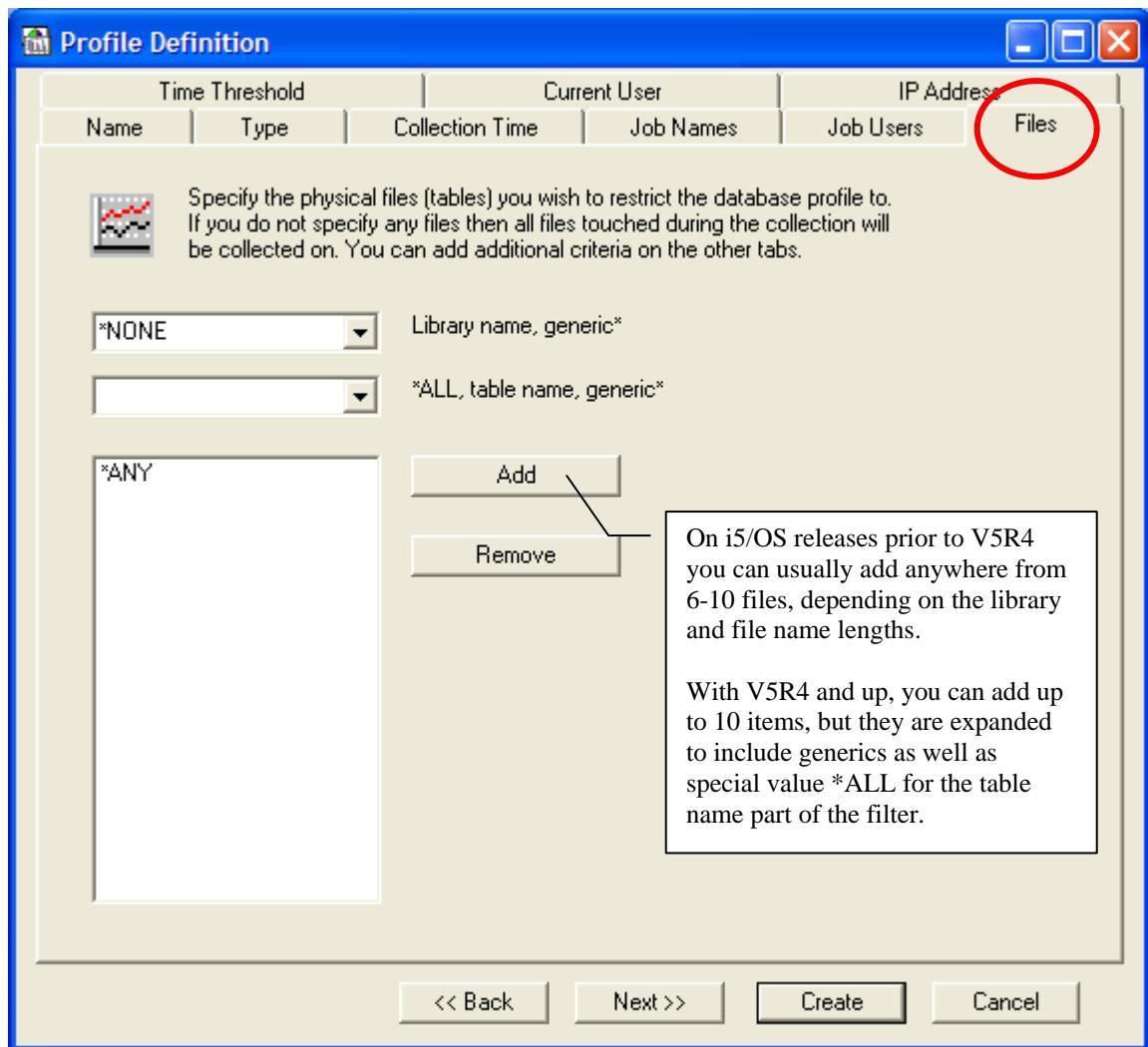
When you are finished with **Job name** option, press **[Next]**.

The **Job users** tab allows you to identify jobs that have a certain job user name. To identify job users a list of actual system users is presented on the **Job users** tab. Note that some jobs, such as the server jobs that run on behalf of IBM's ODBC driver, typically run under QUSER.



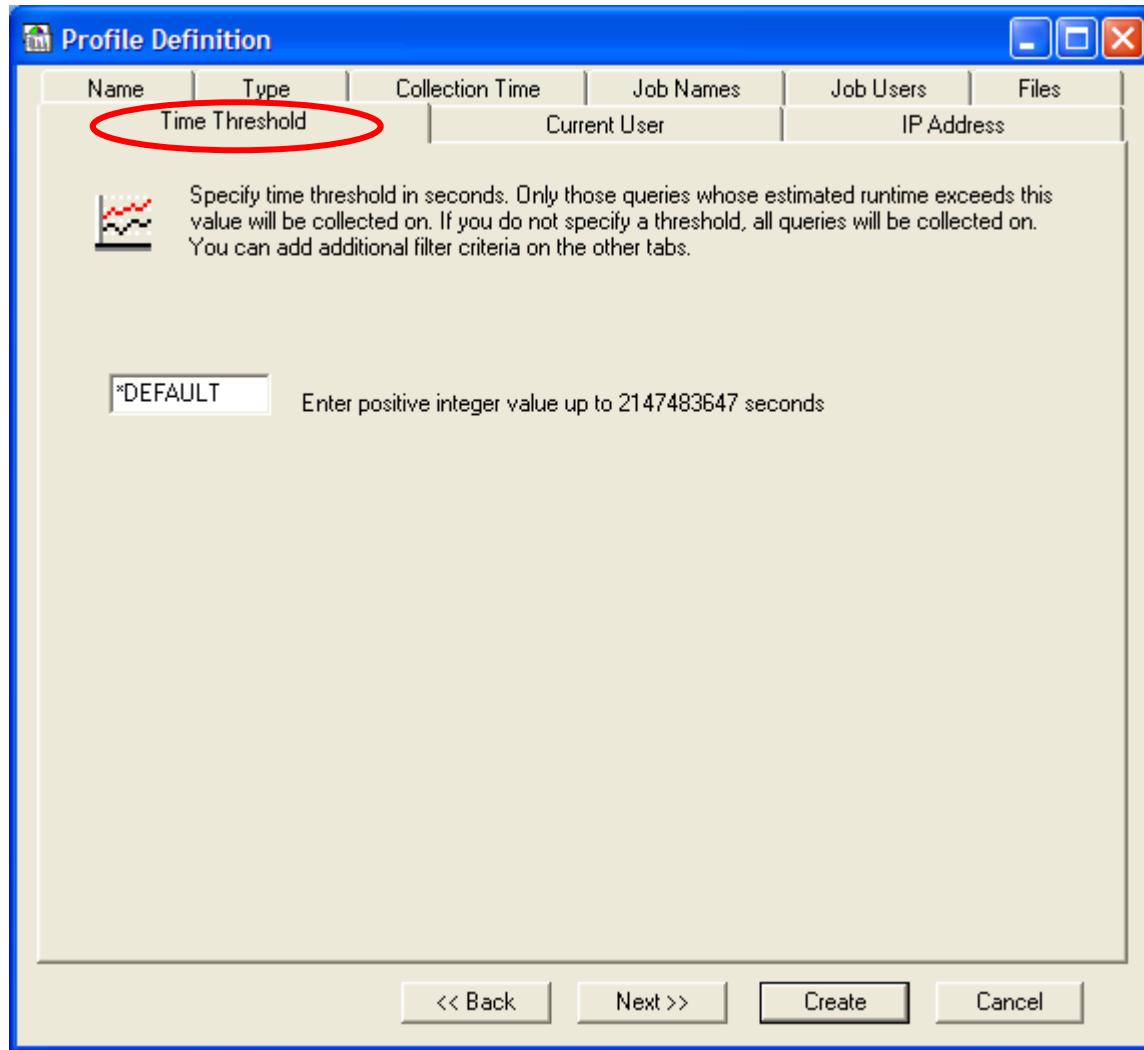
When you are finished with **Job users** option, press **[Next]**.

The **Files** option is especially useful if you want to focus your collection on specific files (SQL tables). For example if you have a large Master file that you know is being queried heavily and want to build appropriate indexes over it, you could narrow down your collection to queries that hit only your Master file. Subsequent index recommendations and query analysis will display results only pertaining to the queries that hit the Master file.



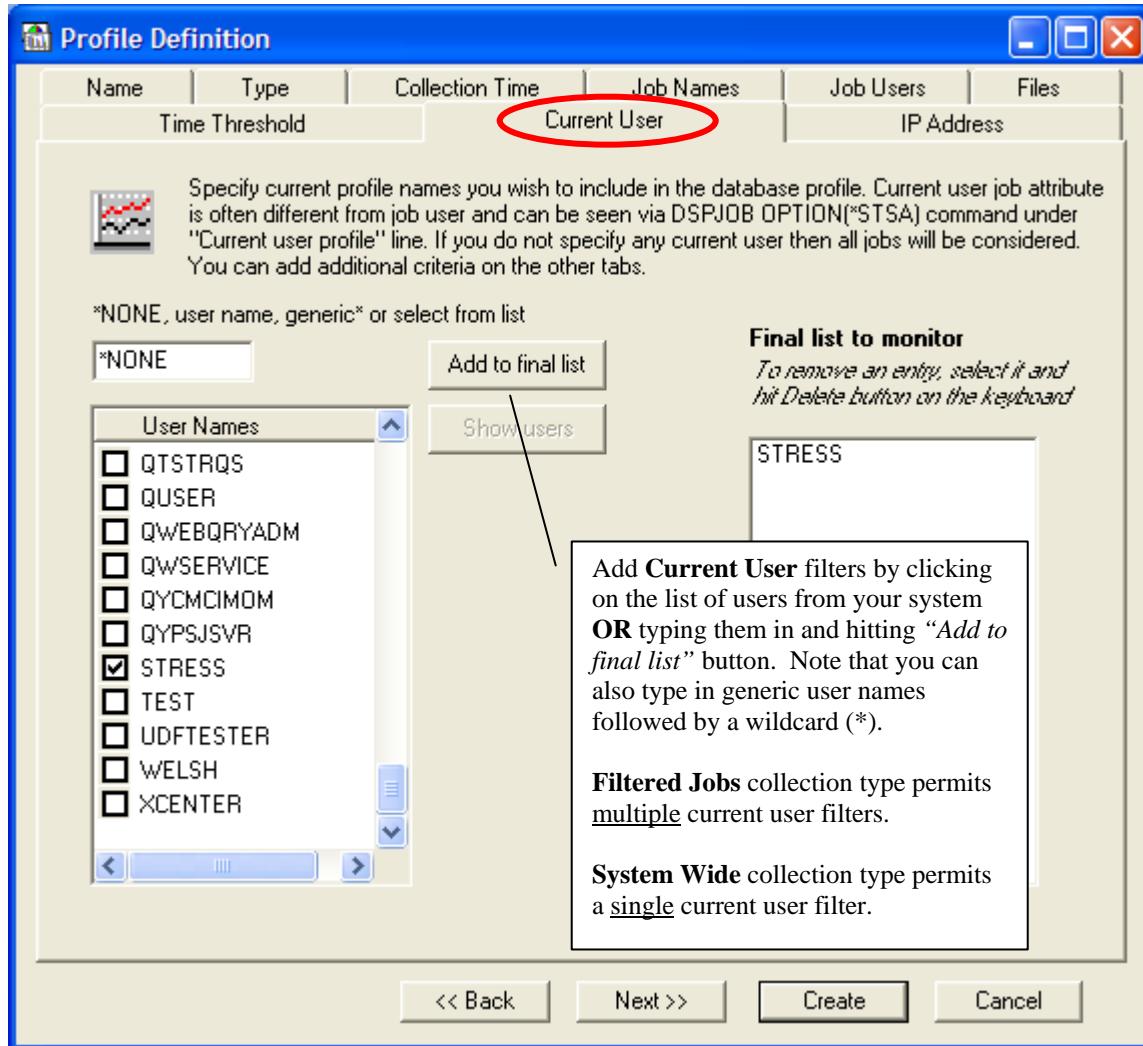
When you are finished with **Files** option, press **[Next]**.

The **Time Threshold** option is especially useful for batch jobs that have long running queries you may want to tune for performance. By specifying a time threshold you will filter out any queries whose estimated run time is less than the threshold specified. Subsequent index recommendations and query analysis will display results only for queries that met your criteria.



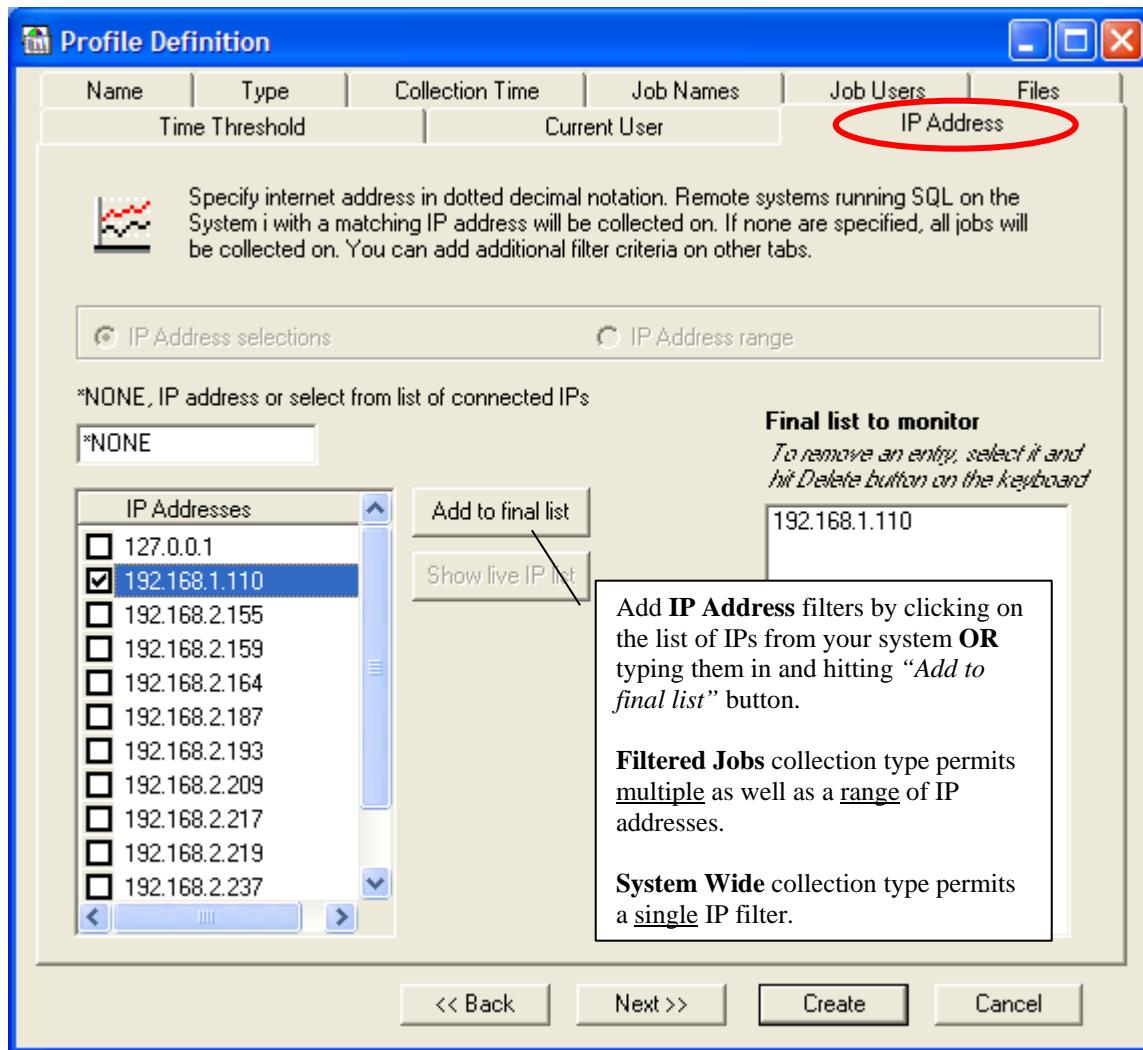
When you are finished with **Time Threshold** option, press **[Next]**.

The **Current User** option is especially useful for collecting on number of IBM host server prestarted jobs. Perfect situations for this type of filter are ODBC/JDBC server jobs (QZDASOINIT, QSQSERVER job names) as well as DRDA server jobs (QRWTSRVR).



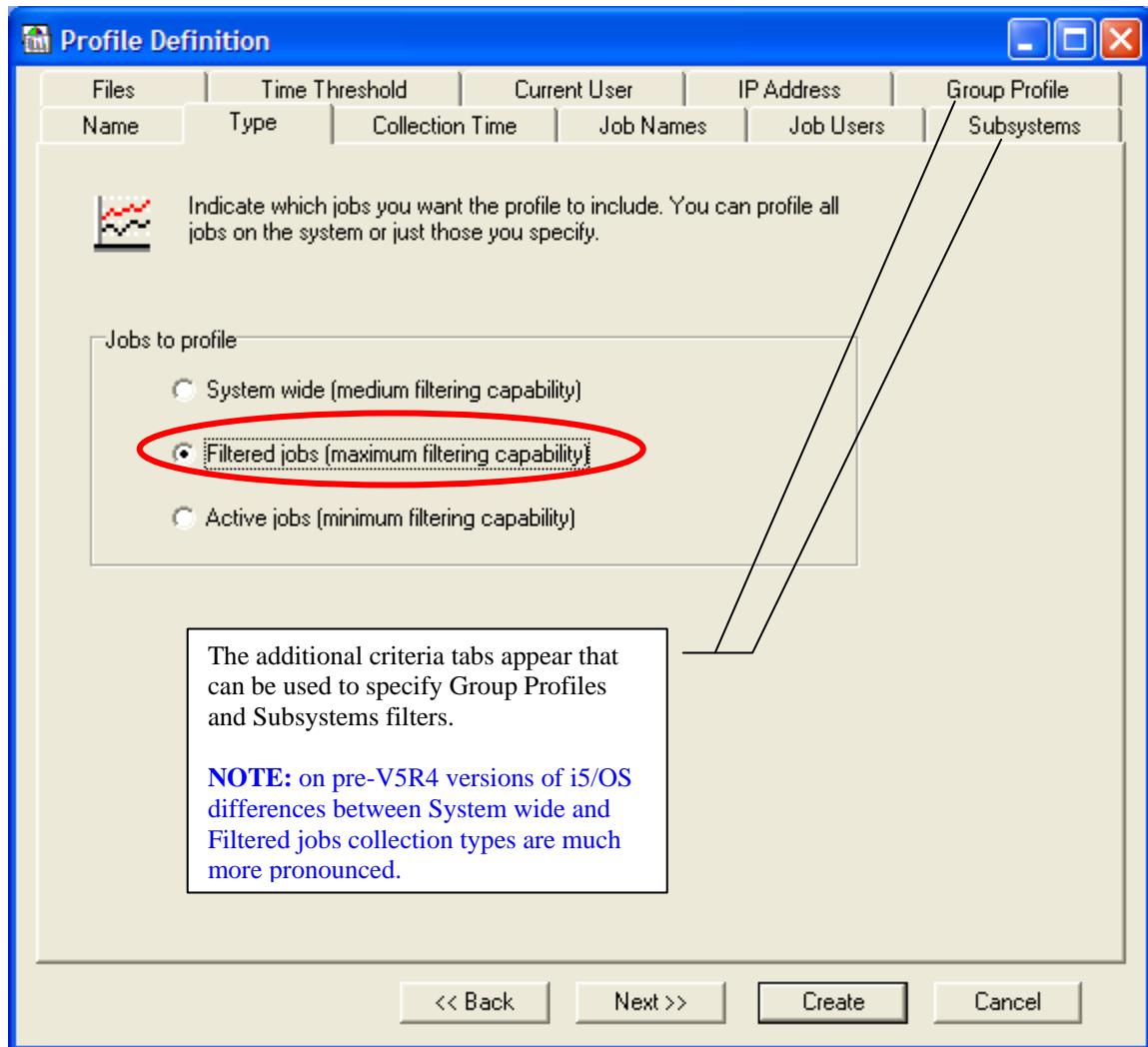
When you are finished with **Current User** option, press **[Next]**.

The **IP Address** option is especially useful for collecting on a specific Web server IP address. Filtered Jobs collection type also permits for IP address range filtering.



When you are finished with **IP Address** option, press **[Next]**.

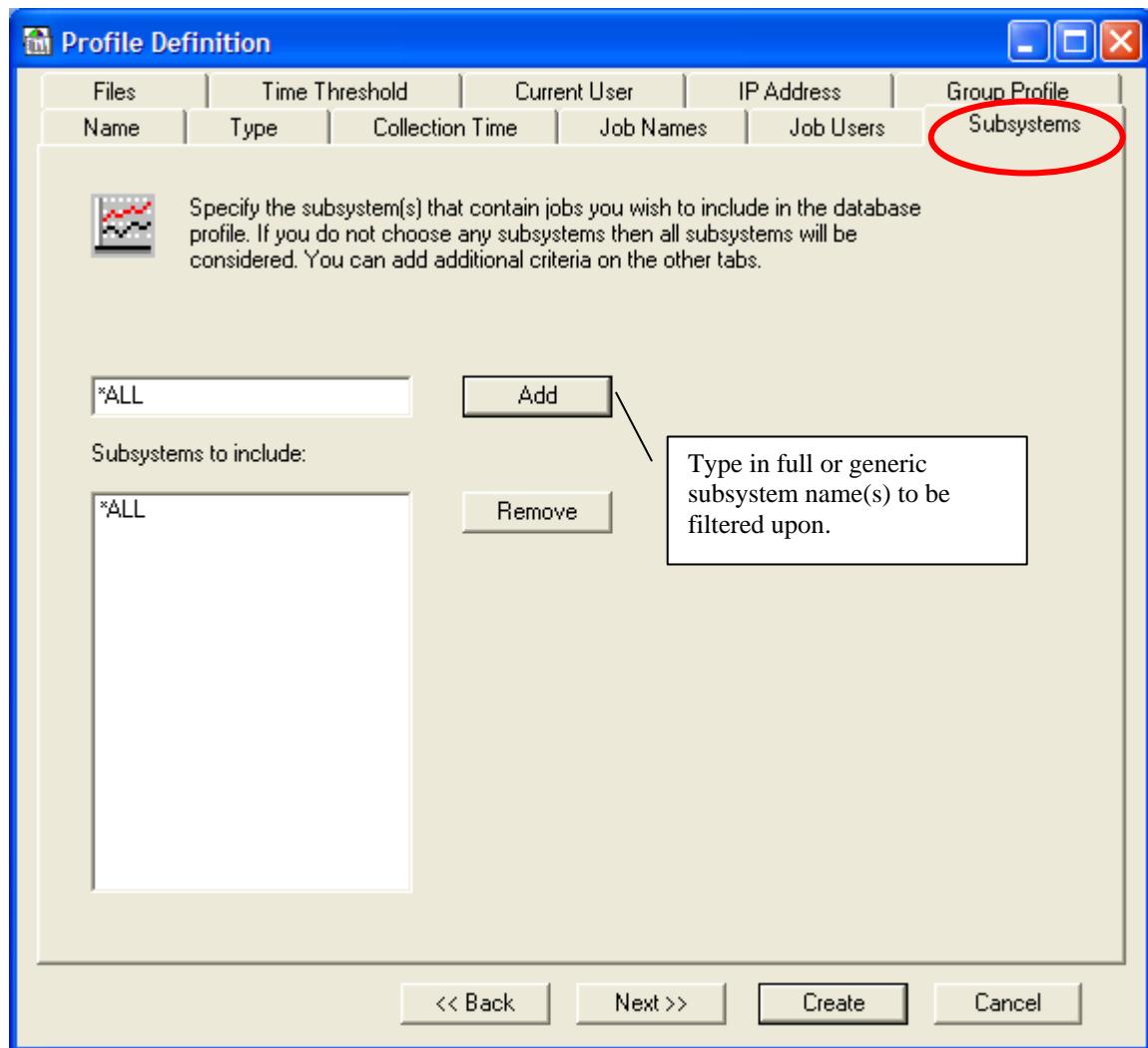
If you choose to ***Filtered jobs*** collection type, you will see additional tabs appear.



If you choose to ***Filtered jobs*** collection type, you should be aware of this prerequisite:

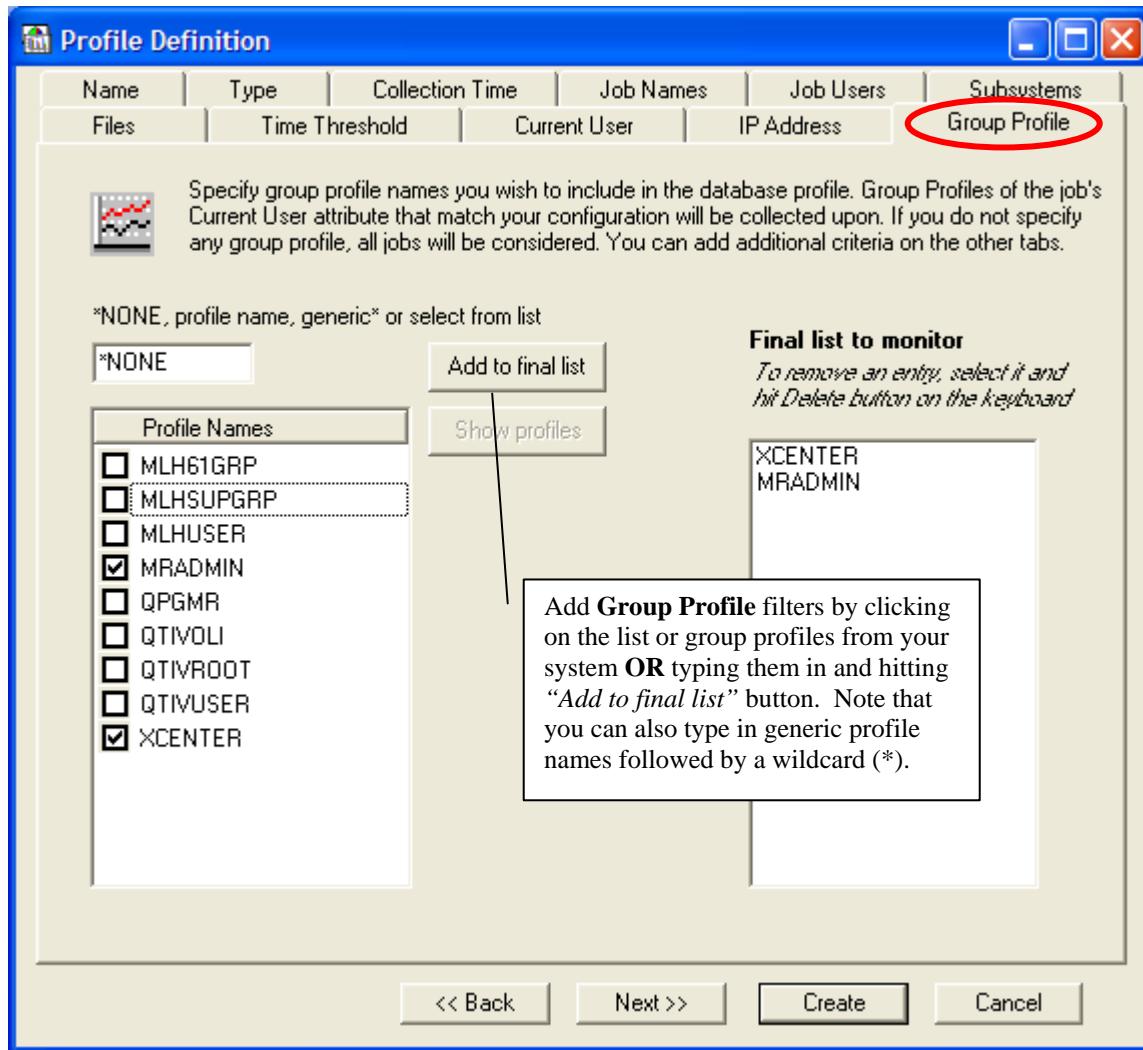
- The first time you choose this option after installing the HomeRun server, any subsystem that might contain a job that would be monitored must be restarted before the monitoring will take effect. (This is because HomeRun registers a program with the subsystem, and the registration is only recognized at subsystem startup time.)

The **Subsystems** tab allows you to identify jobs that run under a certain subsystem. This and other filters are ANDed together so subsystem may help in situations where you may want to collect for a particular job only when running in a certain subsystem.



When you are finished with **Subsystems** option, press **[Next]**.

The **Group Profile** tab allows you to identify jobs whose current user's group profile matches your configuration. This filter is especially useful in organizations whose workload is divided by the group they belong to (i.e. HR or DEVELOPERS)



Once you have all of the information for the database profile properly filled in on all the tabs, press **[Create]**.

Now, you need to wait until the SQL or query activity that you want to analyze is complete. Once the activity you are capturing is complete, end the collection (if you are manually controlling the process). When the last active database profile ends, all data that is buffered or ready for loading is prepared for analysis. The process of preparing data for analysis involves cleaning the data and transforming it into a form that is more suitable for reporting. This process can take several minutes.

3.3.3 Controlling the Usage Tracker's use of resources

Collecting database performance data can be both CPU- and memory-intensive, and can quickly consume lots of disk storage. The process of collecting the data can slow down jobs which are running SQL, and the data collected can get very large. Because of these resource demands at collection time, you must weigh your tuning and analysis needs against the affect the collection activity will have on the system.

To get the most complete picture of SQL activity on your system, you would want to collect profile information for all jobs. Yet, that could well slow down jobs running SQL which does not need to be tuned. It might also unnecessarily consume disk space with information about jobs you do not need to tune.

By choosing to collect profile information on only some subset of jobs, you can significantly reduce the amount of information that you collect requiring less system resources to store and analyze the information. You also affect the performance of only the SQL activity that you are monitoring and not all SQL activity on the system.

You must balance these considerations when deciding how much information to collect.

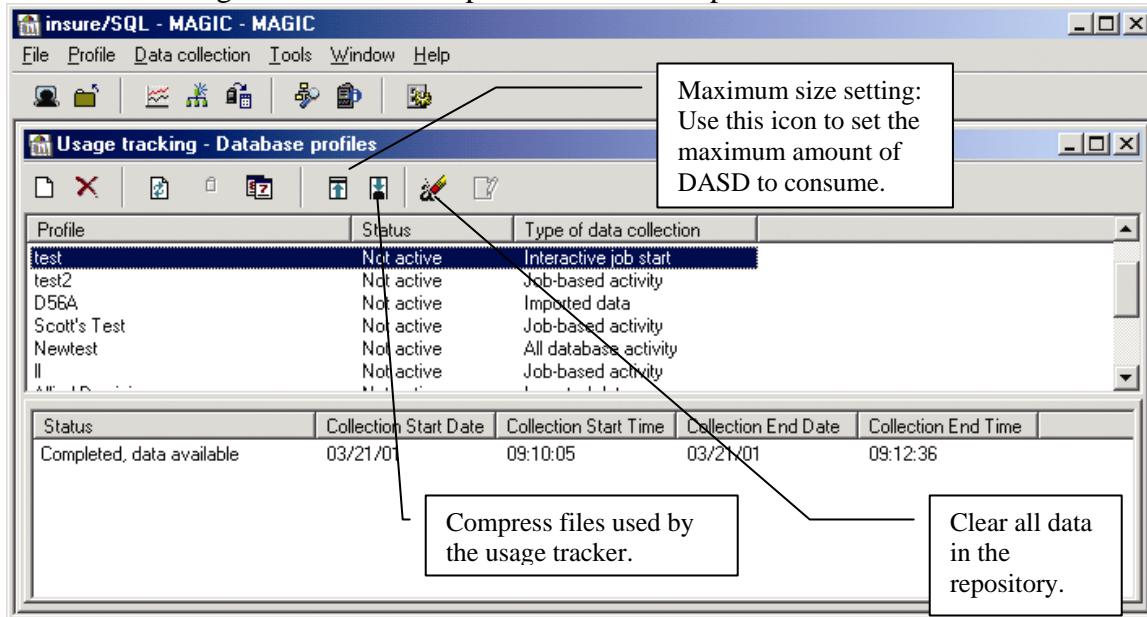
3.3.3.1 Controlling disk space usage

While the most effective way to control space utilization is by profiling a subset of jobs, HomeRun also contains a global setting to control how much space can be consumed by the current set of active collections. If the amount of disk space exceeds this limit, all of the active profiles will automatically be stopped to avoid consuming additional DASD. The maximum can be adjusted based on the amount of space you can allocate to active collections.

If necessary, you can also recover disk space by reorganizing the files used to collect and hold performance information. During normal use, this space is recovered automatically when profiles are deleted. If it is necessary to recover the disk space quickly however, you can use the compress icon on the main profile dialog to recover the space immediately.

Additionally, you can clear all of the data in the repository. This is a much faster method of recovering disk space and ‘starting over’ than deleting each profile or collection individually. This can be done from the usage tracker toolbar or via the CLRREP command in the *{PROGRAM LIBRARY}* library. If you choose to clear the data repository, all profiles that are not scheduled will also be removed since the data associated with them has been deleted.

See the following window for an explanation of each option:



3.3.4 Importing SQL activity information

insure/INDEX and insure/ANALYSIS also allow you to work with database performance information that has been collected without the Usage Tracker.

Imported data must have been collected with the IBM System i™ database monitor, using the command STRDBMON. Data can be imported into the HomeRun toolset either from the same IBM System i™ system that has HomeRun installed, or from a remote IBM System i™. However, if you import from a remote system, you will need an additional license for HomeRun.

To import database monitor data into HomeRun:

1. Contact Centerfield Technology to ensure you have the proper license, if you will be importing data from a remote system.
2. Collect the data using the STRDBMON and ENDDBMON commands. Note the file name and library the database monitor data is stored in.
3. If data is collected on a remote system, move it to the system which has the HomeRun license.
4. Use the IMPPRFDTA command in the HomeRun library to import the data into HomeRun. At this time, you will be able to give this collection profile a name.
5. When IMPPRFDTA has completed, your collection profile will be visible under the new name you gave it when you use insure/INDEX or insure/ANALYSIS.

Note the following restrictions regarding imported collection profiles:

- If the data has been modified or trimmed down after collection, the trimming

should be done using date and time as the criteria

- Real user profile resolution cannot be performed.
- If the source data came from a different system, file long name resolution cannot be performed

3.4 Identifying important indexes with insure/INDEX

After you have successfully captured activity information and it has been prepared for analysis, the profile data will be analyzed automatically and index recommendations captured for that collection. If desired, you can also do more selective analysis or less selective analysis as described in the following paragraphs. **If you do not need to do more selective analysis you can skip the rest of this section.**

insure/INDEX analyzes database activity that has occurred and identifies important indexes that can improve performance. insure/INDEX can focus on certain profiles, libraries, and files. It can also analyze all profiles, libraries, and files in a single pass.

Using a series of performance analysis algorithms, it identifies the set of indexes that you should have on your system. It ranks these indexes so the index that will have the largest benefit is first and the index with the least benefit is last.

To use the index manager feature of insure/INDEX when you want to select different data collections or narrow the search to specific libraries or files (i.e. you do not want to use the default recommendations made when the data collection profile ended):

- 1) Start *insure/INDEX* from the Windows *Start* menu. Select the system you want to manage and press *Connect*. Enter a user profile and password that is authorized to administer your System i system.
- 2) You will be presented with an initial *Welcome* screen. Select *Identify and manage indexes that will improve performance*. This will activate the index manager feature. You can also access the index manager feature by choosing *Tools/ insure/INDEX/Index manager* menu option.
- 3) When prompted with the next wizard window, choose *Identify additional indexes needed on my system*. This will start the Index Advisor wizard interface.
- 4) Choose **[Next>>]**.
- 5) Select *Use the selected profile*. Highlight the database profile that contains the activity you want to analyze. If you choose *Analyze all available data*, the Index Advisor will consider all activity that is currently recorded on the System i regardless of what profile it is part of. Press **[Next>>]**.

-
- 6) Select the libraries or files that you want to analyze. By default, the Index Advisor will analyze the needed indexes for the entire system. Press [**Next>>**].
 - 7) Press [**Analyze**] to begin the analysis. This may take a while depending on the amount of activity that needs to be analyzed. Press [**Save**] when prompted to save the results.
 - 8) You will now see a list of indexes that are needed on your system. If no indexes are listed, your system does not need any additional indexes based on the activity you analyzed. Typically, there will be at least one index listed if this is the first time you used this function.
 - 9) Before creating any of the recommended indexes, you need to decide how you are going to manage changes to your database. insure/INDEX has several built-in capabilities to make this easy. See the following section to understand the options for managing changes.

3.5 Managing changes to your database

You should never make changes to your database, such as index creations, without a set of policies and procedure in place to help manage the change.

insure/INDEX has several built-in features to help you manage change to your database.

Change history

insure/INDEX will maintain a change history of any database changes you make while using insure/INDEX. From this history, you can selectively remove and apply changes or later generate source code scripts for any changes that you made.

Source code output

insure/INDEX can optionally generate source code to remove and apply changes. Using these scripts, you can manage any changes you make with your source control system. You can also make the changes outside of the insure/INDEX interface using standard change management approaches used in your organization.

At this point you need to decide how you are going to manage the changes that you make

to your database.

Do you want to be able to change your database layout directly using insure/INDEX?

Yes

This is the default behavior. Any changes you make will be maintained in the insure/INDEX change history that is stored on the System i and is shared among all users. You can also use a source control approach to track the changes that you make from the user interface.

Note:

Normal System i object security also applies and you will not be able to create any indexes that you are not authorized to create regardless of your policy on this topic.

No

You do not want to make changes directly from within insure/INDEX. Because insure/INDEX does not make the changes, it will not automatically keep a history of all the changes you have made. You will need to manage this yourself.

To change the behavior:

- 1) Select *Tools/ insure/INDEX/Create index wizard* from the menu.
- 2) Press **[Cancel]** on the create index wizard window.
- 3) Select the *Source code* tab.
- 4) Check *Save the index creation script*.
- 5) Check *Save the creation script only*.
- 6) Do not actually create the indexes now. Press *Save as default*.
- 7) Press **[Cancel]** to exit.

You have just disabled the automatic creation of indexes and have set the default option to not create indexes. This information is stored on the workstation, so if you use multiple workstations, you will need to set these options on each of them.

Do you want to generate source code scripts to also track your changes?

Yes

This option will allow you to generate a script with your changes. If you do not want to make changes directly with insure/INDEX, you must choose this option.

To set the default source options:

- 1) Select *Tools/ insure/INDEX/Create index wizard* from the menu.

-
- 2) Press [**Cancel**] on the create index wizard.
 - 3) Select the *Source code* tab.
 - 4) Check *Save the index creation script*.
 - 5) Enter the default library, source file, and source member you want to use.
 - 6) Check *Append to the end of the member if it exists*.
 - 7) Press *Save as default* to save the changes.
 - 8) Press [**Cancel**] to exit.

You have successfully configured your default scripting behavior. This information is stored on the workstation, so if you use multiple workstations, you will need to set this option on each of them.

No

This is the default behavior. Source will not be generated by default. You can always generate source later from the history or change the default on an index-by-index basis.

Using generated scripts

The generated scripts are for use with the System i RUNSQLSTM command. If you are making changes outside of insure/INDEX, you will need to apply changes in this way.

3.6 Creating indexes recommended by insure/INDEX

You are now ready to create the indexes recommended by insure/INDEX.

Before you create the indexes, it is important to know the following.

- Indexes created over large files will take a while to create and may take up quite a bit of storage. Index creation can also consume a lot of CPU. It is normally best to schedule index creation for after hours.
- Once an index is created, it is available for use by the System i query optimizer. You do not have to take any additional steps to make the index available for usage.
- There is no guarantee that an index will be used once it is created. The System i optimizer will choose the best alternative based on the information it has. insure/INDEX recommends indexes that are in the most usable form for the optimizer based on the activity that was profiled.
- insure/INDEX creates SQL indexes, not DDS based logical files.

To create the indexes recommended by insure/INDEX.

- 1) Start *insure/INDEX* from the Windows *Start* menu. Select the system you want to manage and press *Connect*. Enter a user profile and password that is authorized to administer your System i system.
- 2) You will be presented with an initial *Welcome* screen. Select *Identify and manage indexes that will help improve performance*. This will activate the index manager feature. You can also access the index manager feature by choosing *Tools/ insure/INDEX/Index manager* menu option.
- 3) If prompted, choose *Work with existing index recommendations and history*.
- 4) Select *Analysis sessions* in the tree. Select the index advice you generated previously.
- 5) Highlight one or more indexes you want to create. Select *Actions/Create selected indexes* from the main menu, or right-click and choose *Create index*. The index(es) you highlighted will appear in the Create index window.
- 6) You can rename any of the indexes listed in the Create index window by selecting the line, then clicking the selected line once more. To change any other attributes, select an index and press [**Edit**].
- 7) Decide if you want to create the indexes using insure/INDEX or if you just want to generate a script. If you only want a script, select the *Source code* tab and ensure *Save the creation script only* is checked. Verify all of the other source attributes. If you only want a script, proceed to step 10.
- 8) On the *Details* tab, choose the job options for creating the indexes. If you choose batch or schedule, specify the job description and schedule time.
- 9) On the *Details* tab, enter a description to identify this activity in the history list. If you choose an existing description, the history will be added to the existing history.
- 10) Press [**Create**] to perform the action. This will:

-
- Generate the source code if applicable
 - Create the indexes if *Run interactively* was selected
 - Schedule or submit the action to batch
- 11) If you selected *Save the index creation script*, the script is now available on the System i server.
 - 12) If you created the indexes using insure/INDEX, use the history to track status and identify if any errors that have occurred.
 - 13) By selected indexes in the history, you can generate additional scripts or drop and recreate the indexes.

4 Autonomic Database Assistant (AutoDBA)

Welcome to Autonomic DBA, tool that will help you automate database administration tasks.

4.1 Overview

For over ten years, Centerfield Technology has been providing customers with an easy way to find and create indexes to optimize their IBM System i queries. There has never been an easy way to find unused, duplicate, or suboptimal indexes before now however.

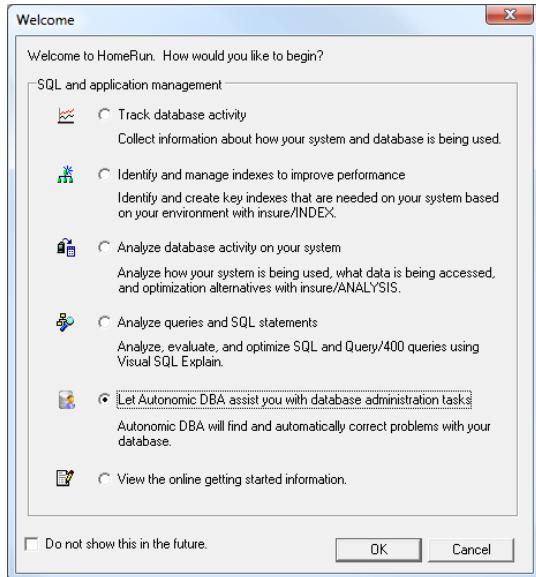
AutoDBA not only has the functionality to suggest indexes which need to be created, but also identify which indexes are not being used and can be deleted or changed to improve performance.

AutoDBA is the first fully autonomic database tool for the IBM System i. What does it mean to be fully autonomic? Autonomic tools should be configurable, run without human intervention, and be fully undoable. Centerfield Technology's AutoDBA has all this plus more. You can configure AutoDBA in many ways and give the tool permission to run tasks without having to manually view data or even click a button. In no time at all your system could start to be optimized without you having to do anything other than turn AutoDBA on. If you are worried about what AutoDBA is doing you can always look in the audit file and even undo actions that either you or AutoDBA have done. The best part about AutoDBA is the new checks and balances that have been put into place with the unused indexes. There are many cases where an index is advised and should be created but then never used again. With AutoDBA you don't have to worry about this.

AutoDBA with the permissions set up correctly (the default is no action) will create indexes that will help your system performance, then if that index is not used again AutoDBA will clean up that index to eliminate the extra overhead.

4.2 Getting Started

Getting started is very easy. When you first start insure INDEX & ANALYSIS you will be prompted on where to begin. Select the default, "Let Autonomic DBA assist you with database administration tasks" (as seen below).



AutoDBA's shipped default is to generate advice about your database every Friday at 7PM. This schedule can be changed to fit your needs. Once AutoDBA has run a "Discovery Phase" on your system, it is time to go through the advice and start optimizing your system. AutoDBA's shipped default is to not perform any action, which forces the user to right click and execute the advised action. For a more detailed explanation on how to get started check out the Getting Started webpage for AutoDBA at <http://www.centerfieldtechnology.com/autodba/gettingstarted.htm>.

4.3 Advice

The screenshot shows the 'Autonomic DBA - All advised' interface. At the top, there are two red boxes: 'All Advised section' pointing to the main grid area, and 'Information on individual advice' pointing to the right side of the screen. On the left, three red boxes point to specific sections: 'Individual advice categories' pointing to the 'All advised' button in the toolbar; 'Done and Undone actions' pointing to the 'View undone actions' button; and 'Advice requiring user action' pointing to the 'View external advice' button. The main grid displays advice items with columns for 'Advice Benefit', 'Times Advised', 'Advice', 'Library', 'File', and 'Member'. The grid shows numerous rows of advice, such as 'Recreate Select/Omit index using DYNSLT DDS keyword' and 'Create index to improve performance'. At the bottom of the grid, there are summary statistics: 'Total advised: 15494', 'Total displayed: 5000', 'High: 1577', and 'Medium: 732'.

In this section you will find all the advised actions that AutoDBA has found. The first time you start AutoDBA you will not see any advice because by default the discovery is configured to run once a week. To see immediate advice you can run the “Discover all” collection in the “Execute now” section (see below for more details) or you can schedule a collection to run earlier than the default. Once the discovery completes, you will see advice in this section. You can choose to view all the advice in the “All advised” section or go into more detail of each type by selecting each category individually. For more detail on each advice scroll all the way to the right and click on the online help link.

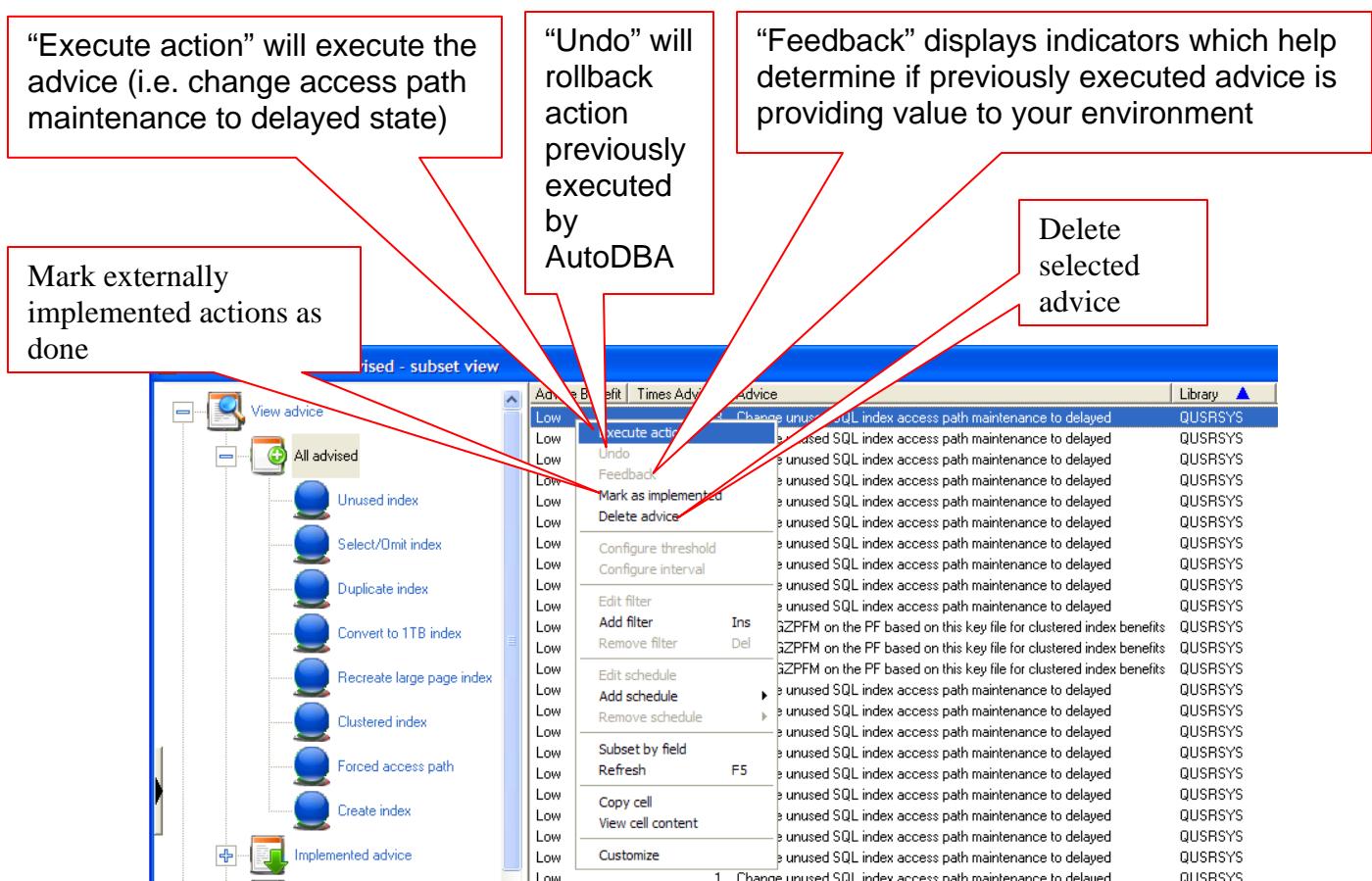
Tip: You can customize data display grid by selecting right-click Customize option. If you'd prefer not to scroll to the right to see online links to additional information for each advice, once inside the Customize window select “Get additional advice info online” column, click on Move Up and hold the Enter key until column is positioned where you want it.

The “All advice” main category has subcategories for each type of advice. You may select to work with all advice or work with a specific subcategory by clicking on one of subcategory items (i.e. Unused index).

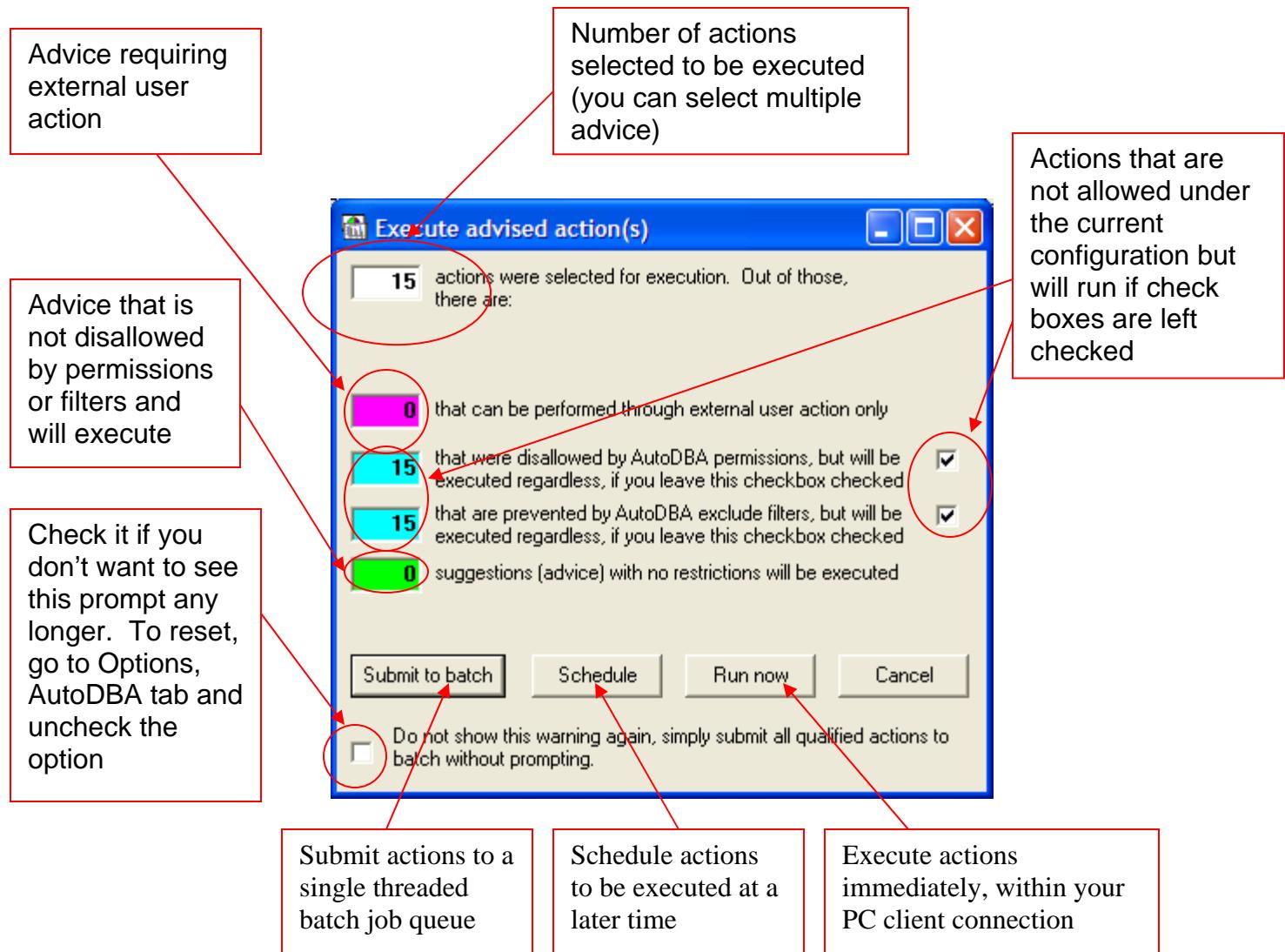
The data grid is initially sorted by Advice Benefit column (priority) as the primary sort column. The grid may employ a secondary sort column as well which is different depending on the advice category chosen (i.e. Days unused for Unused index advice).

Tip: You can sort on any column by clicking on the column heading. You will see a sort indicator indicating current sort order (descending/ascending). Clicking multiple times on the same column header behaves as a toggle switch (DESC/ASC).

Once you review the advice presented in the data grid and are ready to take some action against specific advice, right-click on the advice to display list of available menu options. Some menu options will be disabled (grayed out) as they don’t apply to that specific advice (see image below):



If you choose to manually run the advice by right clicking and selecting Execute menu option, you will be presented with the following prompt for confirmation.



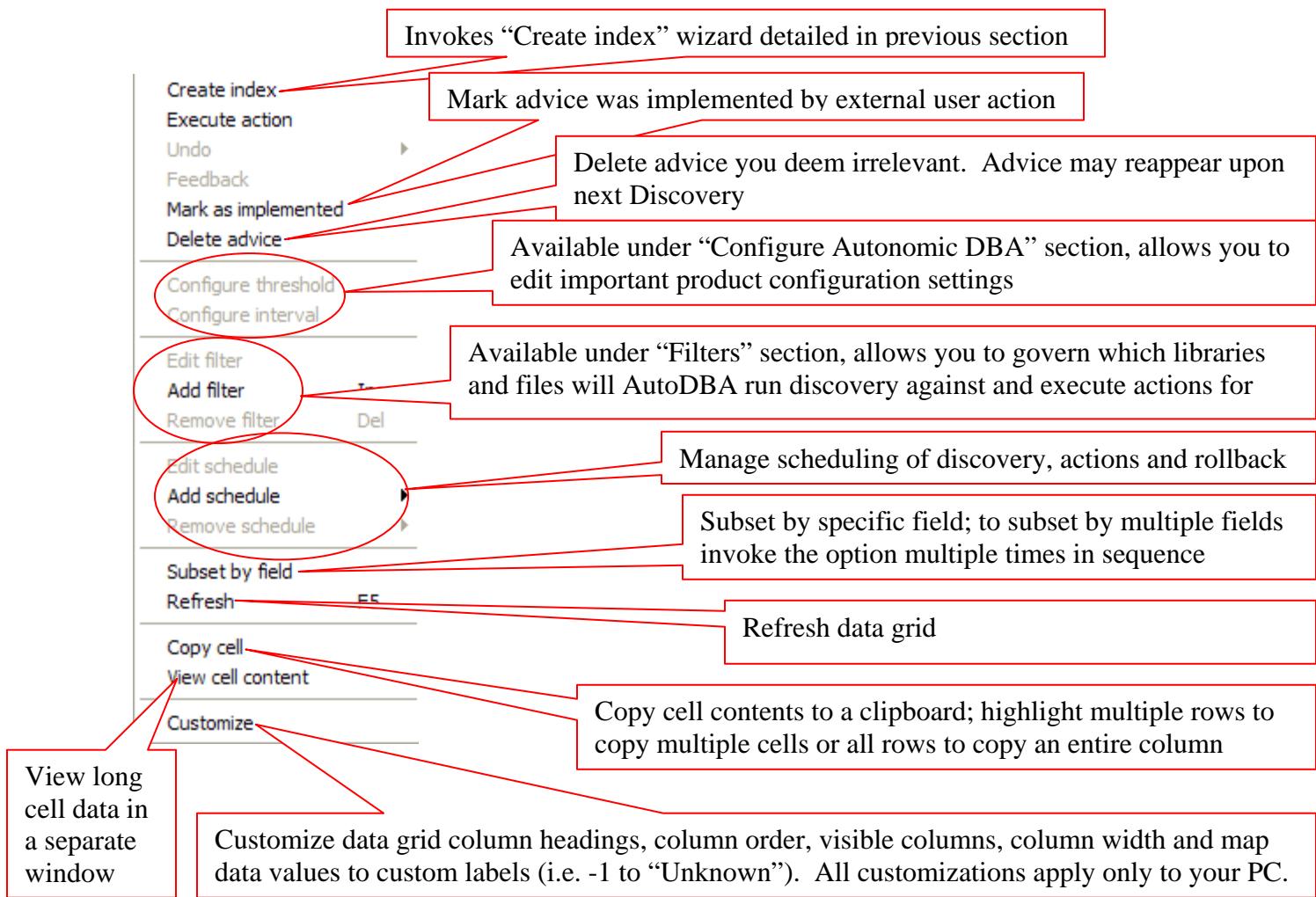
If the advice is to create an index you can also right click and select the dynamically inserted “Create index” menu option which will bring up the index creation wizard traditionally used in insure/INDEX module. The “Create Index” wizard provides a number of options like naming the index in compliance with your naming conventions, specifying a non-data library for index creation, etc. An index created in this manner is still audited and undoable (can be rolled back).

If customizing index options is not relevant to your shop, simply choose the Execute option instead and go with product defaults.

NOTE:

In versions **V5R2** and **V5R3** of IBM's i5/OS you will need to schedule (global) profile collections in insure/INDEX to run on a regular basis so you can gather index creation recommendations, whereas in version V5R4 and higher of IBM's i5/OS this index advice is harvested without a need to collect database monitor data.

4.4 Miscellaneous right-click menu options



4.5 Configuration

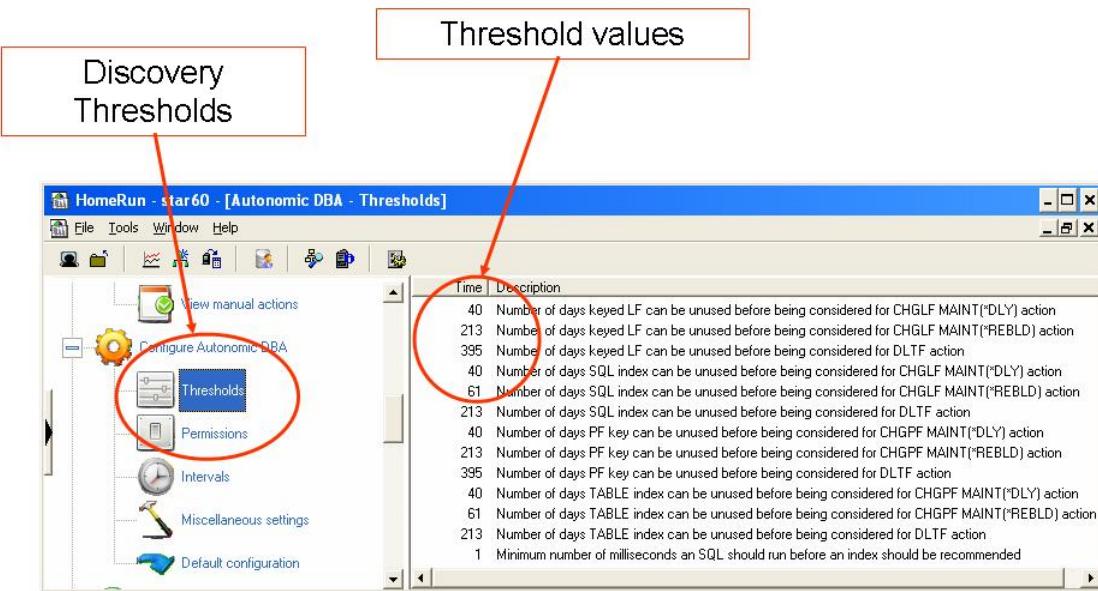
Configuration settings define the way AutoDBA operates. Centerfield ships a predefined set of configuration settings that will work well for a majority of environments, but due to diversity and complexity of large customer environments it may be necessary to further customize the shipped configuration settings.

The following sections discuss how to fine tune Autonomic DBA to work in accordance with your unique environment.

4.5.1 Thresholds

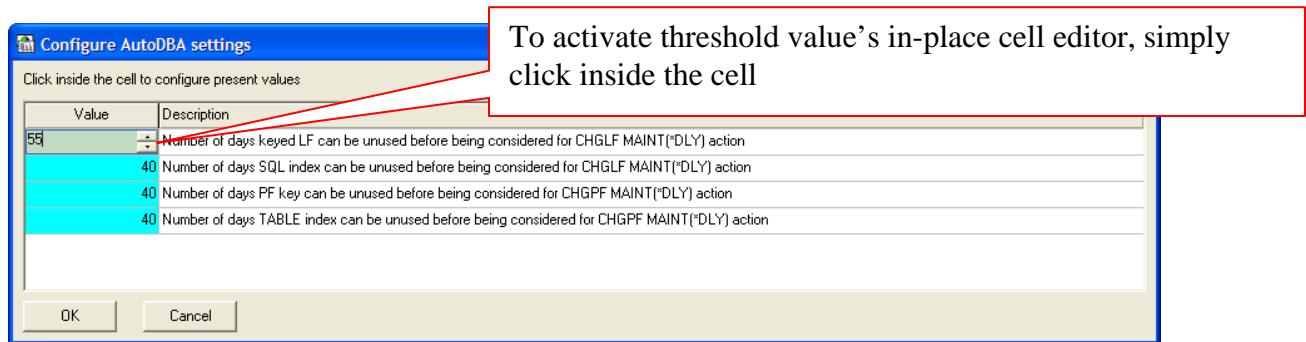
The **sensitivity** of **discovery** is controlled by thresholds. These settings provide control metrics that make it more or less likely that a certain type of advice will be generated. For example, you can decide what should be considered an “unused” index or logical file by adjusting the thresholds. By setting the unused threshold to be fewer days, the number of indexes identified by AutoDBA will increase. By setting the same threshold higher, fewer indexes will be identified as unused.

The Discovery (analysis) phase in AutoDBA is controlled by threshold settings and filters. The thresholds determine how sensitive AutoDBA is to a particular measurement. In the graphical interface the thresholds are configured in a window similar to the following:



By adjusting the threshold values up or down, the number and type of advice recommendations can be changed to fit your environment. It is recommended that the defaults be used until enough information is captured by the Discovery phase to help you decide on new settings.

To bring up threshold values editor window, right-click on one or more threshold values and select the “Configure threshold” menu option.



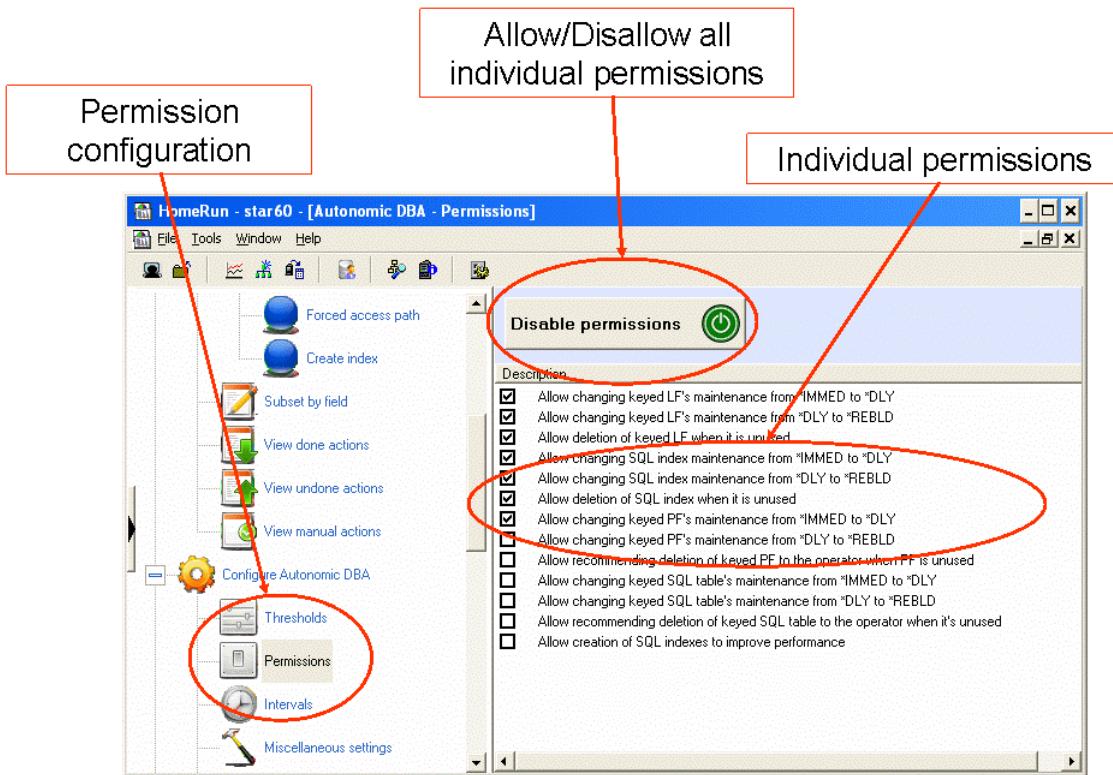
4.5.2 Permissions

As you use the Autonomic DBA and build trust with the advice made by it, you may decide to do away with manual action execution and let AutoDBA execute certain actions for you automatically. First step to the enablement is to set the permission switches appropriately.

Permissions are global switches that determine if automatic **actions** can be taken. There is a primary switch that determines if any actions can be taken and there are individual switches for each type of advice. You can for example, turn on the overall switch and allow one of the rules to be automated while still keeping the other actions manual.

Tip: Think of the “Disable permissions” button as the big **OFF** switch. If you set it, all individual permissions are disabled as well (default). This gives you a single spot to control whether **any** actions are executed by AutoDBA background job.

To allow any automatic action in AutoDBA, permissions must be set to allow them. By default no permissions are set which means no automated actions are taken. To set permissions simply check the boxes in the GUI to allow the tool to implement advice.



4.5.3 Intervals

Intervals control how often certain actions are performed by Autonomic DBA background jobs. We recommend against changing them unless advised by Centerfield Technology support personnel.

4.5.4 Miscellaneous settings

Miscellaneous settings control non-critical behavior of AutoDBA.

For example, AutoDBA may find that a certain physical file or SQL table is not being used at all. The inherent restriction built into the AutoDBA is **not** to delete these objects, even if permitted to do so by the user. Instead, AutoDBA will send a message to QSYSOPR message queue to inform system operator that physical file is not used and can be deleted. If you'd rather not see this message in system operator message queue, you can disable the setting in this section. Diagnostic message to this effect will still go the AutoDBA joblog.

4.5.5 Filters

Discovery filters control which objects AutoDBA can analyze and generate advice for.

NOTE:

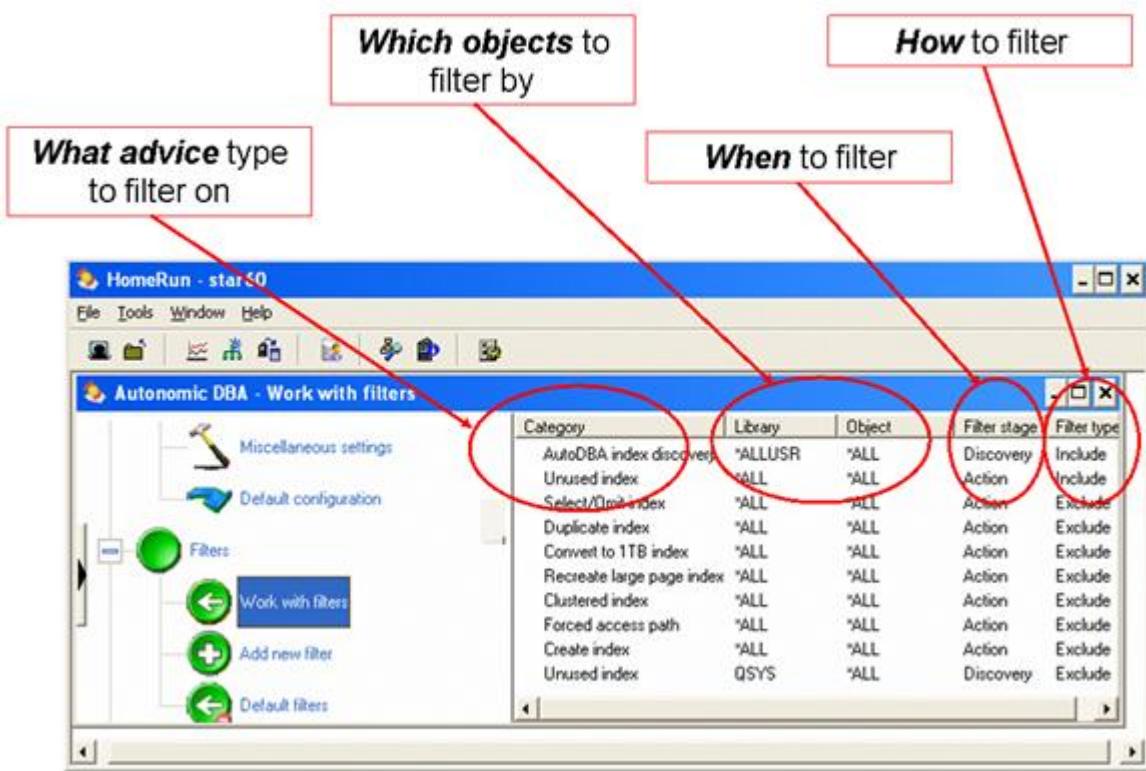
Present focus of Autonomic DBA is index management and **discovery** filters apply to **physical files** (tables) **only**. Advice however, will be generated for any keyed access paths over these physical files (keyed LFs, SQL indexes and non-unique physical file keys).

The default discovery filter is to consider all user objects for any index advice (unused, suboptimal, missing index ...) and is designated by “Index advice discovery” umbrella category. We recommend you keep this default for discovery phase as there are no negative connotations to the discovery phase.

Action filters work much like discovery filters in that they control which objects can have automated actions performed on them. By default, all objects are excluded from actions. To allow automatic implementation of advice you must define “include” filters to grant AutoDBA the authority to consider objects for automatic action execution.

The Discovery and Action phases of AutoDBA can be directed toward or away from specific libraries and database files. This is accomplished through the use of object filters. An object filter can be an “include” or “exclude” type. By default, the Discovery phase tends to be inclusive in the sense that many objects are normally examined to determine what types of optimizations are possible. On the other hand, the Action phase tends to be exclusionary in that by default all objects are excluded from automatic actions.

In the graphical interface, both types of filters are managed by a window that looks similar to the following:



Four factors fully define an object filter. When defining or editing a filter, you decide:

1. What type of advice or action to control.
2. Which objects you would like to include or exclude.
3. Whether the filter applies to discovery (analysis) or to action (execution).
4. Whether the filter includes or excludes objects.

It is recommended that you use the default filters when initially using AutoDBA, and over time adjust them to suite your goals.

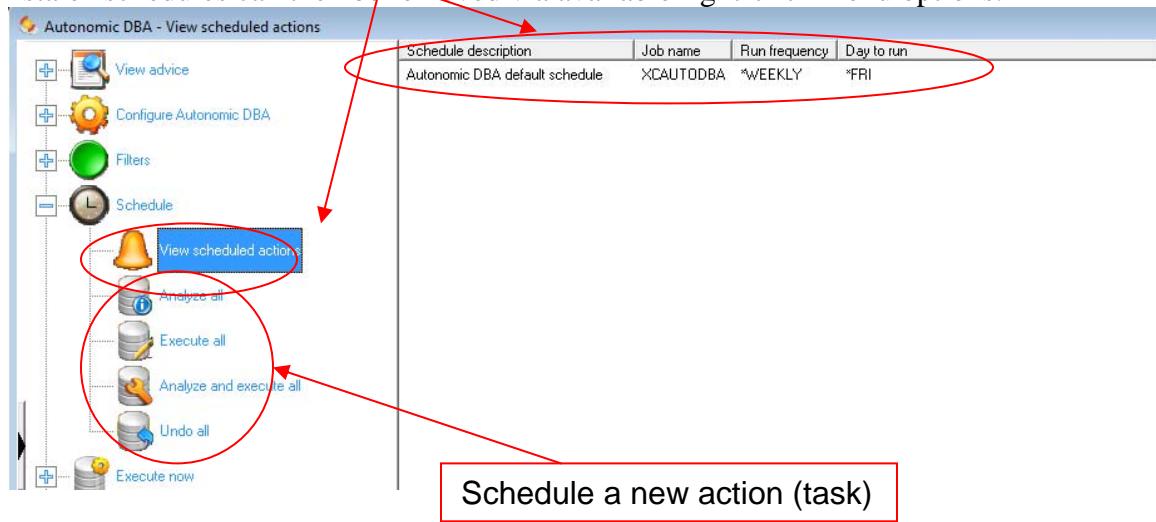
A common scenario is where a user develops trust into AutoDBA generated advice and defines Action-Include filters to allow AutoDBA to perform actions automatically against objects specified in the defined filter.

4.5.6 Schedule

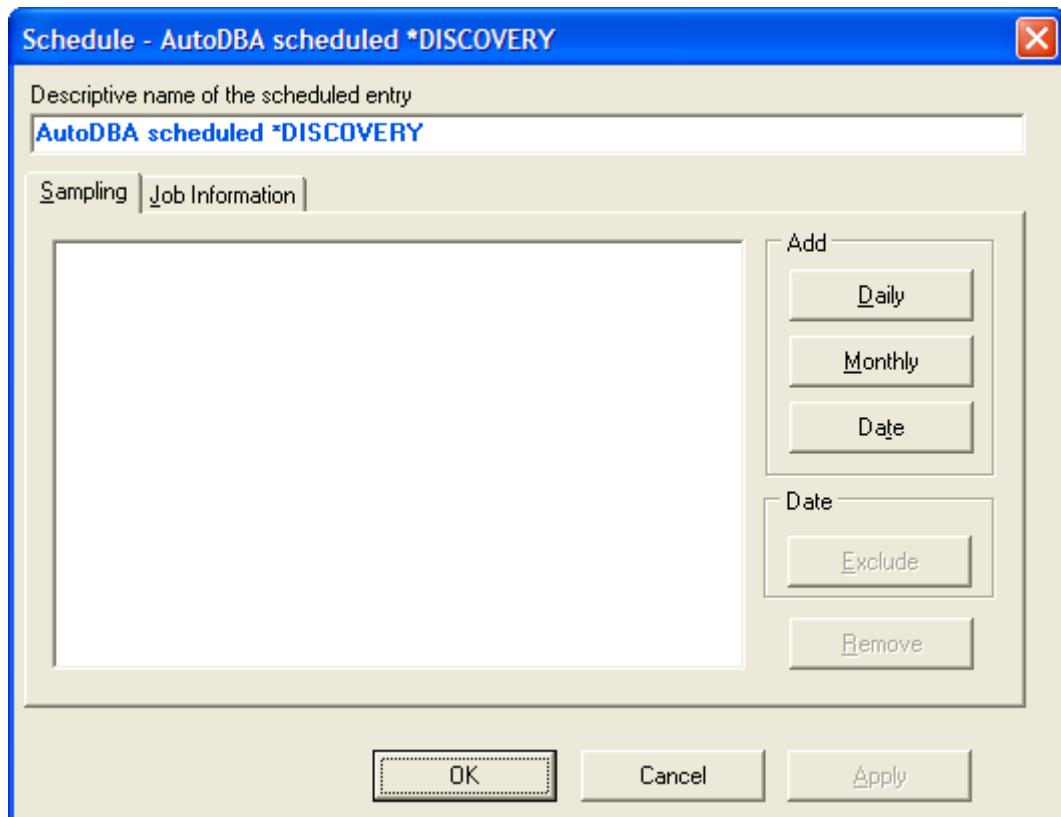
This is the portion of the product that allows you to schedule when AutoDBA will run a specific task like “Discover all” or “Execute all actions”. You can view the currently scheduled tasks in the “View scheduled actions” section or you can schedule a new task

View the currently scheduled actions
by clicking on the action you would like to schedule.

You can also see history of scheduled actions that have already taken place. These "stale" schedules can then be removed via available right-click menu options.



When scheduling a new action you will see the following screen or one like it, where you can schedule the action either daily, monthly or by date.

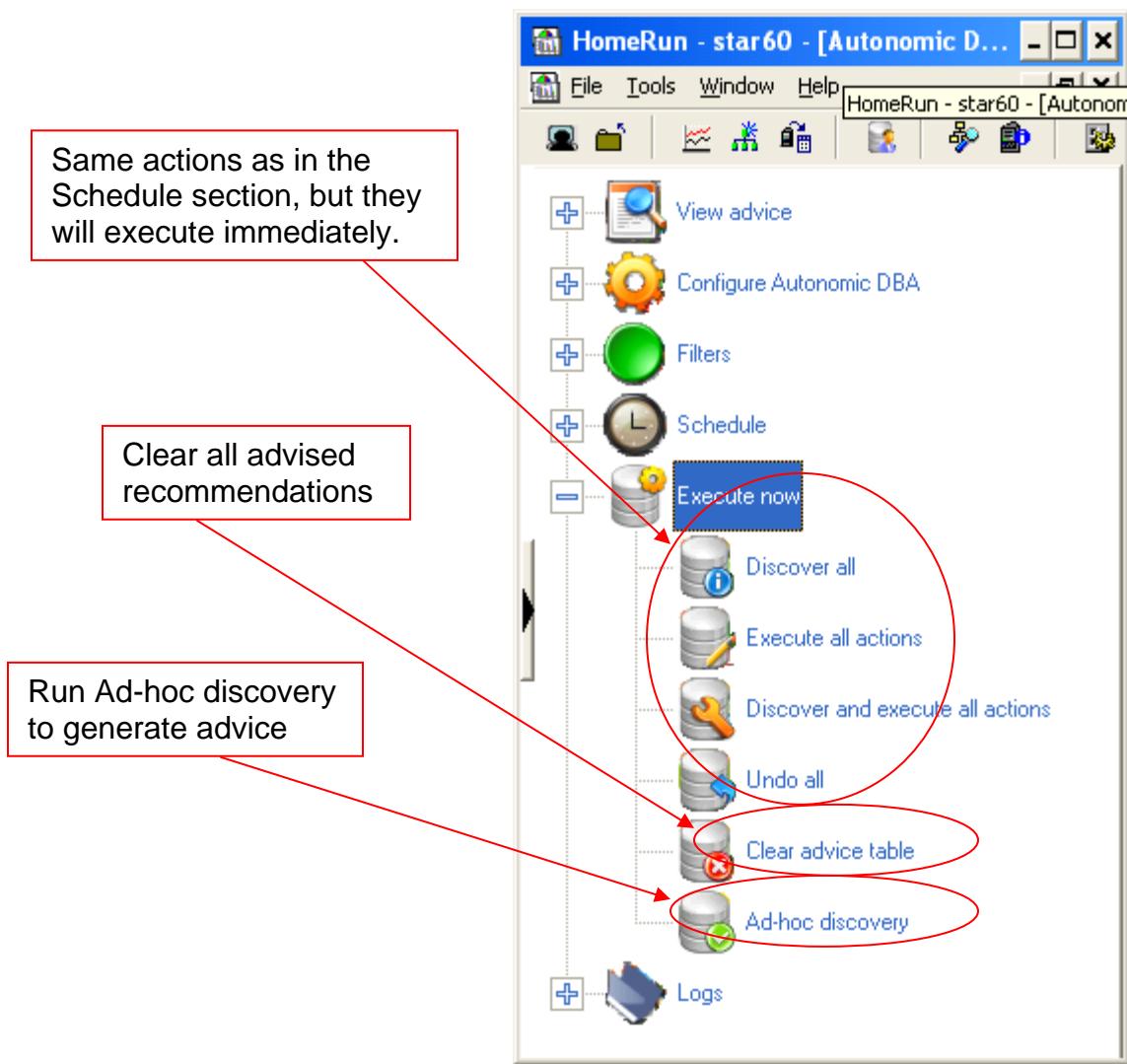


NOTE:

Running “Execute all” right out of the box (fresh install) will not do anything as the shipped values for the permissions and filters do not grant AutoDBA authority to execute anything. Make sure you set permissions and define some Action-Include filters if you want to the Execute all command to have any effect.

The “Execute now” section is the same as the Schedule section of the product except AutoDBA will run the action immediately instead of waiting for a scheduled time (i.e. inside the current job). Keep in mind the time of the day and how busy your system is before running any action immediately.

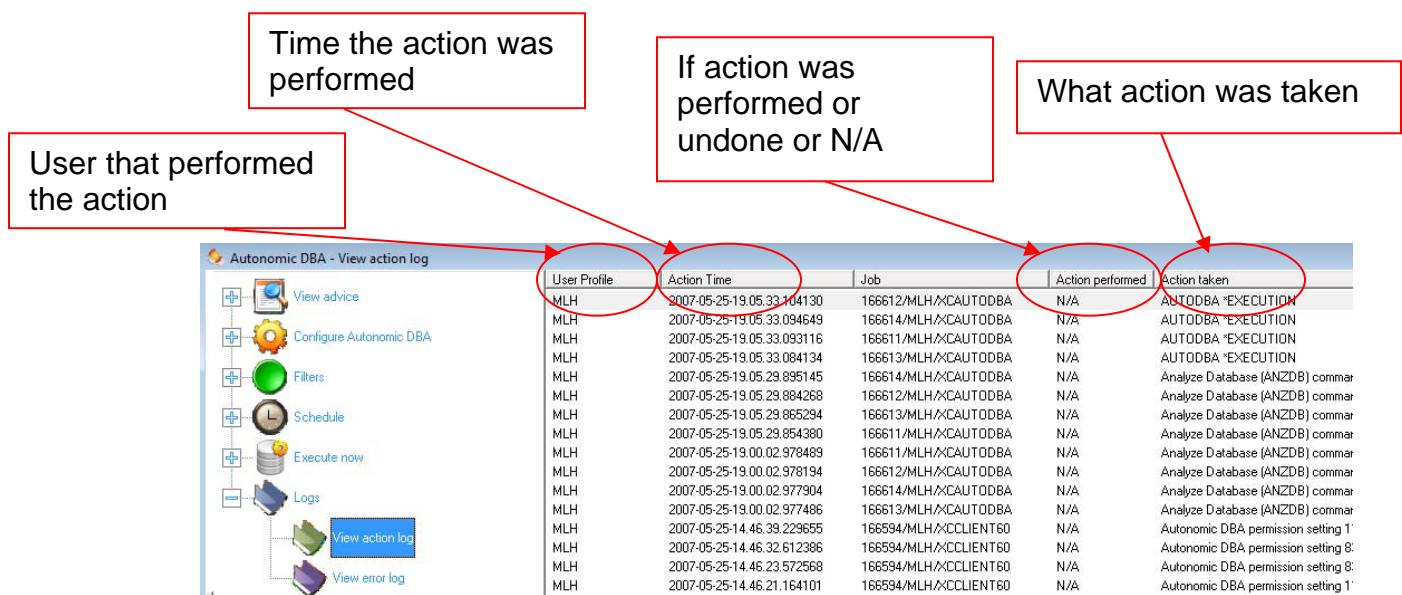
You can also clear the advice table and start over and run an ad-hoc discovery on the fly. You will be prompted with a message verifying that you want to proceed with the action selected.



Tip: When clearing advice table, we strongly encourage you **do not** check “Clear executed advice as well” checkbox, as this will disable you from undoing any previously executed actions.

4.7 Logs

The logging feature is a place for users to go to see what AutoDBA and other users have been doing on the system through Centerfield Technology's product. You can even undo actions that you deem unnecessary or incorrect. This is a great place to audit AutoDBA and the users of this tool.



You can right-click and undo an action whose "Action performed" is set to "Yes".

Normally, the error log will be empty. We do expect the discovery phase to occasionally encounter issues analyzing certain objects (i.e. no authority to the object, object exclusively locked at the time etc.). These types of errors will be handled and logged to the error log, and AutoDBA will proceed to analyze next object in the list.

If you see other types of errors persistently showing up in the error log, contact Centerfield Technology support. It would be helpful if you can provide the support personnel with a copy of the XCERRORLOG file and the spooled joblog of the job listed in the error log message.

4.8 Privileges

Autonomic DBA offers very powerful functionality which system administrators may want to share with other IT staff by redistributing the Centerfield PC client installation file.

If your business rules clearly delineate personnel that can and cannot perform certain functions, you may need to customize AutoDBA privileges to allow or disallow certain users from specific AutoDBA functionality within the PC client.

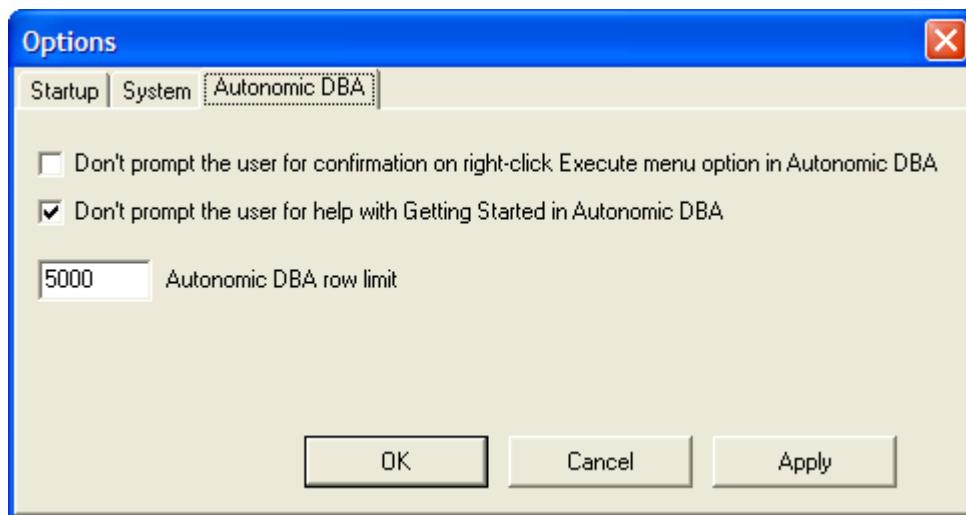
To accomplish that click on **Tools | Privileges** to invoke privileges editor window. In the window, scroll down to “Use Autonomic DBA assistant” and check or uncheck desired privileges for select users.

For more details on how to define specific privileges refer to section “Securing the use of insure/SQL tools”.

4.9 Options

Autonomic DBA has number of options that control the PC client behavior. To view and alter these options, invoke the options menu **Tools | Options | Autonomic DBA** or click on the shortcut icon  directly.

You will be presented with the following screen:



A key option in this display is the row limit, which controls how many advice items a user can see in the advice grid at any one time.

Be aware that the PC grid component will slow down dramatically if the row limit is higher than 10000, so exercise care when setting this limit.

We believe our default is quite sufficient for most users.

5 Support

Centerfield Technology Inc. is committed to providing our customers with support as problems or questions arise. Support includes minor enhancement releases and upgrades which might be necessary as OS/400 and i5/OS releases become available.

5.1 Contact Information

Internet

Web support pages are maintained at www.centerfieldtechnology.com. Updated documentation, how-to's, FAQs, and a list of known problems will become available as needed.

E-mail

To contact technical support by email, send email requests to support@centerfieldtechnology.com.

Fax

To contact technical support by fax, send your request to **(888) 908-3073**.

Phone

You can call 507-287-8119 for technical support. Telephone support is available from 8:00 AM to 5:00 PM CST.

5.2 Additional Information

- Visit Centerfield Technology's HomeRun web site www.centerfieldtechnology.com to find the latest software patches for HomeRun. Follow the support link from the home page.
PTFs listed in this document are current as of the date this document was released. There are times where IBM releases PTFs applicable to the functions our software uses after the document release date. Accordingly, we strongly recommend you visit our website periodically and verify that all listed PTFs are applied on your system.

6 Built-in Tools

6.1 Using Visual SQL Explain

Visual SQL Explain allows you to quickly understand the implementation of an SQL statement. It makes statement tuning easier by visually showing the actions that the System i query optimizer will use to implement a particular statement and by putting all of the influencing controls at your fingertips. It provides access to the standard SQL debug messages provided by the optimizer and it exposes important statistics related to the implementation path.

6.1.1 Visual SQL Explain plan constructs

The following plan constructs are used when implementing an SQL statement.

6.1.1.1 Sequential (arrival) file access methods

-  Table scan – Reads all records selecting the ones that match the selection criteria.
-  Parallel pre-fetch – Reads all of the records in parallel using 2 or more tasks.
-  Skip sequential processing – Uses a bitmap to selectively read the matching record, reading records in the physical order.
-  Parallel skip sequential processing – Uses a bitmap and 2 or more tasks to selectively read the matching records.
-  Encoded vector index load – Processes an EVI index while creating a bitmap.
-  Parallel encoded vector index load – Processes an EVI index using 2 or more tasks while creating a bitmap.

6.1.1.2 Key field (index based) access methods

 Key row positioning – Use an index to selectively read records from a physical file returning the records in order.

 Key row selection – Use an index to read all records from a physical file in order.

 Index only positioning – Use an index to return the fields selected by the query.

 Index only selection – Read the entire index to return the fields selected by the query.

6.1.1.3 Record processing methods

 Group by – Process the GROUP BY clause record grouping.

 Hash group by – Process the GROUP BY clause using a hashing algorithm.

 Sort processing – Process the ORDER BY clause using a memory based sort algorithm.

 Distinct processing – Process the DISTINCT clause, returning only unique records.

 Record selection – Apply the WHERE clause selection after reading a record.

 Union – Combine the results of two or more SELECT statements.

6.1.1.4 Join algorithms



Nested loop join – Join records by reading a record from the first file then finding a matching record in a second file until all matching records are processed in the first file.



Hash join – Join records by first reading all records and organizing them into subsets. Then join matching rows within each subset.

6.1.1.5 Intermediate result processing



Temporary index build – Build an index because an index is required and a suitable index does not exist.



Temporary results file – Build an intermediate file that contains the results up to this point. The rest of the query can operate against these results.



Temporary file – Build a temporary file to hold the results of a view or other complex source file.



Bitmap creation – Build a bitmap to selectively process records.

6.1.1.6 Visual SQL Explain plan constructs reference



Encoded vector index load

Processes an EVI index while creating a bitmap.

This is a good alternative if you do not have SMP capabilities on your system. To influence encoded vector index loading:

Specify the level of parallel processing that is allowed using either the QQRYDEGREE system value, HomeRun, or the CHGQRYA command.



Index only positioning

Index only positioning returns a row or set of rows ordered by a key field. Index only positioning only reads part of an index in a manner similar to a program that uses keys to find a particular record.

Unlike ***key row positioning***, it does not do random I/O to access the record in the physical file.

This makes index only positioning the fastest method for accessing an index. This method has the benefits of table scan processing and the benefits of index processing.

See ***key row positioning*** for implementation tips.



Index only selection

The entire index is read and any selection criteria that references the key columns of the index are applied using the information in the index.

Unlike ***key row selection***, it does not do random I/O to access the record in the physical file.

This makes index only selection the fastest method for accessing an index when the entire index must be processed. This method has the benefits of table scan processing and the benefits of index processing.

See ***key row selection*** for implementation tips.



Key row positioning

Key row positioning returns a row or set of rows ordered by a key field reading. Key row positioning only reads part of an index in a manner similar to program that use keys to find a particular records.

Key row positioning requires a traditional binary radix index.

It is a good alternative for the following cases.

- When a small number of rows will be returned from a file
- When the physical ordering of the records in the file are ordered using the same or a similar order as the key fields in the index.
- When the positioning can be used for join processing and either ORDER BY or GROUP BY processing.
- An existing index can be used instead of a dynamic index build over a large file.

Optimizing index access

- Index access is most efficient when it can do ***index only*** processing.
Criteria for index only processing:
 - All selected fields from the file are contained in the index
 - None of the fields are null-capable
- Specify OPTIMIZE FOR n Rows. If a small number is specified for ‘n’, the query optimizer will tend to choose a keyed access method more often. If a large number is chosen for ‘n’, the optimizer will attempt to use a arrival sequence access method if possible. In general it is best to choose a value for ‘n’ that matches the number of rows you actually expect to be returned from the SQL statement.
- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will allow bitmap based-processing and give the iSeries better statistics about data in the file.
- Remove or modify an ORDER BY clause.
- Change the availability of hash join processing.
- Use the command RGZPFM to organize the physical file to match this index.



Key row selection

This access method requires an index. The entire index is read and any selection criteria that references the key columns of the index is applied against the index.

The advantage of this method is that the table is only accessed to retrieve rows that satisfy the selection criteria applied against the index. Any additional selection not performed through the key selection method is performed at the table level.

The key selection access method can be very expensive if the search condition applies to a large number of rows because:

- The whole index is processed.
- For every key selected from the index, a random I/O to the table occurs.

This is better than key row positioning if every record in the index matches the selection criteria.

Normally, the optimizer will choose to use table scan processing when a search condition applies to a large number of rows. The optimizer only chooses the key selection method if less than 20% of the keys are selected or if an operation forces the use of an index. Options that might force the use of an index include:

- Ordering
- Grouping
- Joining

In these cases, the optimizer may choose to create a temporary index rather than use an existing index. When the optimizer creates a temporary index it uses a 32K page size. An index created using a CREATE INDEX statement normally uses only a 4K page size. The optimizer also processes as much of the selection as possible while building the temporary index. Nearly all temporary indexes built by the optimizer are select/omit or sparse indexes. Finally, the optimizer can use multiple parallel tasks when creating the index. The page size difference, corresponding performance improvement from swapping fewer pages, and the ability to use parallel tasks to create the index may be enough to overcome the overhead of creating an index. Table scans used for building of temporary keyed access paths.

If key selection access method is used because the query specified ordering (an index was required) the query performance might be improved by using one of the following combinations of pre-compiler parameters to allow the ordering to be done with the query sort.

ALWCPYDTA(*OPTIMIZE), ALWBLK(*ALLREAD), and
COMMIT(*CHG or *CS)

ALWCPYDTA(*OPTIMIZE) and COMMIT(*NONE)

Optimizing index access

- Index access is most efficient when it can do *index only* processing.
Criteria for index only processing
 - All selected fields from the file are contained in the index
 - None of the fields are null-capable
- Specify OPTIMIZE FOR n Rows. If a small number is specified for ‘n’, the query optimizer will tend to choose a keyed access method more often. If a large number is chosen for ‘n’, the optimizer will attempt to use a arrival sequence access method if possible. In general it is best to choose a value for ‘n’ that matches the number of rows you actually expect to be returned from the SQL statement.
- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will allow bitmap based-processing and give the iSeries better statistics about data in the file.
- Remove or modify an ORDER BY clause.
- Change the availability of hash join processing.
- Use the command RGZPFM to organize the physical file to match this index.



Parallel pre-fetch

Read, using multiple I/O tasks, all of the records from the source table optimizing disk and memory utilization.

If the percentage of rows selected is greater than 20%, this method is more efficient than index processing. On the other hand, if a small percentage of data is selected and the file is large, this access method is not optimal.

Parallel pre-fetch is a good alternative for the following cases.

- A majority of the records in the file will be selected
- There is sufficient main memory with limited contention available for a larger portion of the file in memory at any given time
- The file is distributed across multiple physical disk units. If the file is distributed across multiple disk units the system can perform many physical disk I/O's at the same time which will reduce overall elapse time of the query.

See **Table scan** for information on how to influence parallel pre-fetch implementation.



Parallel skip sequential processing

Uses a bitmap and 2 or more tasks to process records in the order they physically reside in the file, eliminating records that will not be used in the result set as they are read. This approach is preferred to a table scan as it is more efficient.

Parallel skip sequential processing is a good alternative for the following cases.

- A majority of the records in the file will be selected
- The file contains a small number of records
- You have enough main memory available to allow all of the tasks to run efficiently

Influencing Parallel skip sequential processing

- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will improve skip sequential processing efficiency and give the iSeries better statistics about data in the file.
- Specify the level of parallel processing allowed using either the QQRYDEGREE system value, HomeRun, or the CHGQRYA command.
- See **Table scan** for information on more techniques.



Parallel encoded vector index load

Processes an EVI index while creating a bitmap using 2 or more tasks.

This is a good alternative if you have SMP capabilities on your system. To influence encoded vector index loading:

- Specify the level of parallel processing allowed using either the QQRYDEGREE system value, HomeRun, or the CHGQRYA command.



Skip sequential processing

Read records from a file in the order they physically reside in the file eliminating records that will not be used in the result set as they are read. This approach is preferred to a table scan as it is more efficient.

Skip sequential processing is a good alternative for the following cases.

- A majority of the records in the file will be selected
- The file contains a small number of records

Influencing skip sequential processing

- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will improve skip sequential processing efficiency and give the iSeries better statistics about data in the file.
- See **Table scan** for information on more techniques.



Table scan

Reads the entire file using the physical order of the records. This optimizes disk access as it minimizes random I/O.

If the percentage of rows selected is greater than 20%, this method is more efficient than index processing. On the other hand, if a small percentage of data is selected and the file is large, this access method is not optimal.

Optimizing table scan

- A table scan is most efficient if the optimizer can implement data space selection. This is the lowest level of selection the system can perform and can greatly minimize the CPU requirements to select or reject rows. The following situations allow the optimizer to use data space selection:
 - the fields referenced in the WHERE clause should come directly from the table
 - literal values in the WHERE clause should have the length and precision of the data in the table
 - comparisons with packed decimal or other numeric types can only take advantage of data space selection if the table was created with an SQL CREATE TABLE statement
 - character fields cannot be variable length
- Specify OPTIMIZE FOR n Rows. If a large number is specified for 'n', the query optimizer will tend to choose arrival sequence more often. If a small number is chosen for 'n', the optimizer will attempt to use a keyed access method if possible. In general it is best to choose a value for 'n' that matches the number of rows you actually expect to be returned from the SQL statement.
- Create an encoded vector index (EVI) for each of the fields used in the WHERE clause for this file. This will improve table scan efficiency and give the iSeries better statistics about data in the file.
- Create a traditional binary radix index for key fields used for ORDER BY, GROUP BY, or WHERE clause conditions. This will give the iSeries an index based option to access the file

instead of a table scan. This should be done in cases where less than 20% of the rows in the file will be used in the result set or if these key fields are used often in several queries.

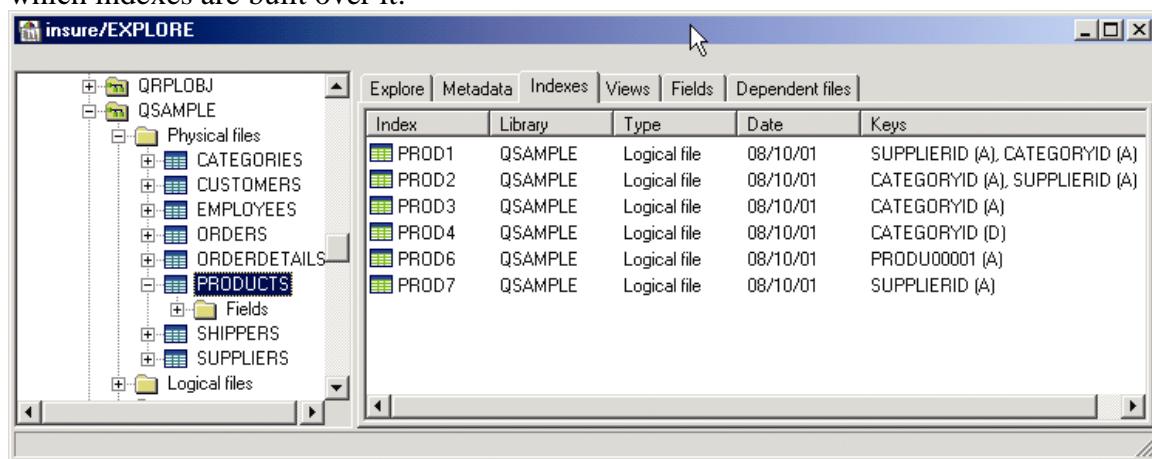
- Add or modify an ORDER BY clause.
- Specify a field from that has an index to bias the optimizer toward index based access.
- Specify a field from another file in the query to bias index access for that file and table scan for this file.
- Specify a field from several files to bias the iSeries to use a sort rather than index access for ordered access to the data.
- Allow I/O or CPU parallel processing. Table scan processing benefits from both I/O and CPU parallel processing.
- Change the availability of hash join processing.

6.2 Using Database Explorer

Database Explorer provides access to database information so you can quickly browse for needed information:

- Browse System i libraries, files, views, indexes
- Browse metadata associated with these objects.
- View field definition information.
- View relationships between files and fields.
- Sort columns in ascending/descending order by clicking the column heading

Database Explorer gives you a familiar, hierarchical tree interface in its left pane, and displays details in the right pane. Navigating through the tree on the left gives you access to details about each item. In this example, the user has navigated to the file PRODUCTS in library QSAMPLE, allowing details to be shown about the file, such as which indexes are built over it:



To access Database Explorer, choose the *Database Explorer* option from the *Tools* menu.

7 Appendixes

7.1 Introduction to database performance management

This section introduces database performance management for query and SQL activity and gives you a basis for the methodology and supporting software provided by the HomeRun toolset. It is intended to give you the basic information you need to optimize your database for SQL and query activity.

The HomeRun tools address database optimization and performance management above and beyond traditional system level performance management. As such, the performance information in this section complements traditional system performance tuning information. In some cases, your database usage and tuning will influence system level performance, such as the use of SMP.

This information applies to you if you have any type of SQL or query activity on your system. Some common environments are:

- Data warehouse or decision support environments
- Query/400 environments
- OPNQRYF environments
- ERP and application environments where embedded and dynamic SQL is used in System i programs
- ODBC and client/server environments or applications
- Internet environments and applications which access System i data using SQL

HomeRun:

1. Enables very straightforward process for managing performance.
2. Identify how your database is being used
3. Provide and maintain the optimal set of resources to support its usage.

7.1.1 Effective performance management

The following items are important to know in order to implement effective performance management. Read through the topics that follow and keep them in mind when designing, optimizing, or diagnosing your database and associated applications.

SQL and query activity

All SQL and query activity on the System i uses a feature of the operating system called the query optimizer. As the name suggests, it is responsible for optimizing access to your database data.

The types of activity that use the query optimizer include:

- Any application that uses SQL
- Query/400
- OPNQRYF
- Data analysis, data warehousing tools
- Client/server applications
- Internet access to System i data
- Client Access
- STRSQL, RUNSQLSTM
- DRDA
- ODBC (IBM, HiT, StarQuest, Wall Data, others)
- JDBC
- CLI

The query optimizer takes your query or SQL statement and determines the optimal way to perform that query based on the resources and information it has available.

To optimize your database and resolve performance issues you will normally do one or more of the following:

- Give the optimizer additional resources it needs (indexes, implementation options, data layout, storage, etc.)
- Eliminate unneeded resources that get in the way (indexes)
- Give the optimizer additional information
- Modify the SQL or query to a form that is more suitable for the optimizer

The System i optimizer will adapt to its environment

The query optimizer on the System i will adapt to its environment. Influencing factors are:

- hardware (main storage, # disk drives, CPU)
- system activity (configured and actual)
- amount of data in the file
- query specific conditions
- release/PTF level of the System i
- SMP and parallel processing

-
- Available indexes
 - Each system must be tuned for its unique environment
 - What's good for one system can be bad for another

This means:

- Every environment is different and should be tuned for its unique configuration and usage.
- Test and production systems differ. This means you need to manage performance in a production environment and need to make changes and manage these changes effectively. HomeRun has many built-in features to help.
- What's good for one environment may not be good for another and may actually cause problems.
- You want to tune for the activity that is occurring in the environment. You do not want to tune for all activity that could occur, only the activity that is actually occurring.
- You want to maintain the proper set of resources for your environment. Too many resources, such as indexes, can cause problems. Not enough resources, such as indexes and memory, can cause problems. You need to identify the optimal set of resources for your environment and its usage.
- As your database grows, your activity changes, or your hardware configuration changes, it is necessary to re-optimize your database for this new environment.

HomeRun makes it easy to identify, analyze, and tune for your unique activity.

7.1.2 Understanding an SQL statement's implementation

SQL and most query products let you specify what data you want, how you want it formatted, etc. However, the way the data is actually retrieved is taken care of by the query optimizer, and is hidden from the user.

In order to diagnose performance problems and fine-tune your environment, it sometimes becomes necessary to look "*under the covers*" to see what is happening.

This requires:

-
1. The ability to look inside the black box
 2. The ability to understand what you see
 3. The ability to take a corrective action

HomeRun helps you with all three of these requirements. The Visual SQL Explain tool and its associated documentation are primarily available for this purpose.

7.1.3 Managing change in a production environment

It is very important to manage any changes you make to your database in a production environment. As you add, delete, and change indexes and file attributes, you need the ability to track those changes and the ability to apply and remove those changes easily.

Some of the important reasons for tracking these changes are:

- Application modifications - Periodically you will need to install newer versions of applications and other software you use on your system. During this process, you will want to remove any indexes you have created to ensure the changes install properly.
- Operations - Your operations staff perform routine maintenance and backups. You want to make sure your changes are tracked and properly managed.
- Disaster recovery - You will want to keep a record of changes in order to recover from a disaster or system failure without starting over from scratch.

HomeRun has several features to help you manage change:

- Built-in change history management
- Integration with source control
- Security to control who can make changes to your system

7.1.4 Understanding side effects

Sometimes side effects occur. For example, say you have two SQL statements. You create an index for the first statement and it runs 20% faster. You then create an index for the second statement. It runs 30% faster, but has the side effect of slowing the first statement down by 50%.

This is a case where the second index was a good resource for the second statement but was a bad resource for the first statement.

To help avoid the problem, there are a few things that you can do.

- Tune the entire database, not individual SQL and queries, as much as possible.
- Tune only high impact individual SQL statements. Do not create every index that might ever be needed.
- Take advantage of the database and program level controls such as “allow copy data” (ALWCPYDTA) and blocking parameters to help the optimizer choose the right alternative.
- Take advantage of SQL syntax such as ‘OPTIMIZE FOR n ROWS’ to give the optimizer additional information it needs.

7.2 System-wide installation impacts

HomeRun changes some settings on your IBM System i™ system at installation time that can affect your current system configuration and activity. The HomeRun modifications that may have system-wide impacts are limited to:

- Changing system values
- Ending currently active Centerfield servers, jobs, and monitors
- Configuring network attributes
- Modifying subsystem configurations
- Adding exit points

This section also details information on HomeRun's use of the following IBM System i™ features:

- TCP/IP usage
- Job scheduler usage
- Authorities on installed libraries
- CL Commands and APIs used
- Programs which adopt authority

7.2.1 System Values

The HomeRun toolset helps you handle issues like performance, security, and usage tracking which requires special system privileges. Because the product installs software that performs privileged operations, it requires certain system values that control authority and application privileges to be set to certain values. The following table shows the list of system values and a description of the HomeRun installation requirements.

QALWUSRDMN	This system value is checked at installation to ensure that the value is either *ALL or that it contains the <i>{PROGRAM LIBRARY}</i> library as one of the libraries that allows user domain objects. HomeRun requires this setting because it stores and directly accesses user space objects in the <i>{PROGRAM LIBRARY}</i> library. If the system value is not set properly for HomeRun, the installation support modifies the setting to include the <i>{PROGRAM LIBRARY}</i> library.
QALWOBJRST	This system value is checked at installation to ensure that the value is either *ALL, or that it at least allows system state objects (*ALWSYSSTT) and objects that adopt owner authority (*ALWPGMADP) to be restored. HomeRun requires this setting because it restores objects with these attributes into the <i>{PROGRAM LIBRARY}</i> and <i>{DATA LIBRARY}</i> library. If the system value is not set properly for HomeRun, the installation will fail.

QUSEADPAUT	This system value is checked at installation time to ensure that the system does not restrict the list of user profiles that are allowed to modify programs to use adopted authority. HomeRun requires this because depending on the system program temporary fix (PTF) level, this system value can also control users that are allowed to run programs that adopt *OWNER authority. If the system value is not set properly for HomeRun, the installation will fail.
------------	--

HomeRun references several other system values at run time to determine information about your system configuration. The following are other system values that are referenced at run time:

- QACGLVL
- QDATE
- QDAY
- QYEAR
- QMONTH
- QSRLNBR

7.2.2 Servers, Jobs, and Monitors

You need to ensure that the objects that will be replaced by the HomeRun installation and the resources that are being modified by the installation are not used or locked by other jobs on the system prior to starting the product installation. HomeRun creates and replaces objects in the *{PROGRAM LIBRARY}* and *{DATA LIBRARY}* libraries. You should ensure that there are no objects locked in these libraries prior to installing the software. However, even if you do not check these libraries before you start the installation, the HomeRun installation checks that some of the common jobs that can hold critical locks or cause other problems are always ended before letting the installation continue.

7.2.3 Database Monitor and HomeRun

Several features within HomeRun build on top of the IBM System i™ database monitor support (STRDBMON command). The HomeRun installation ends the database monitor if it is actively collecting system-wide activity. It ends the monitor to help ensure that there are no locks on files within the *{DATA LIBRARY}* library.

7.2.4 Subsystems and Work Management

The HomeRun server software handles:

- Managing and processing incoming requests from the administrative console software
- Enforcing user access policies configured by the administrative console software
- Scheduling of any activity requested by the administrative console
- Maintaining data collection information

The HomeRun server uses its own work management configuration to control how

HomeRun client jobs are started. At installation time or using the HomeRun CFGSVR IBM System i™ command, you define the subsystem that HomeRun activity should run in. By default the Centerfield server will run in its own subsystem defined by the XCSBS80 subsystem description in the program library. It is highly encouraged that you use this subsystem to isolate the Centerfield server from other system work. At installation time, an autostart job entry is put into the QUSRWRK subsystem. This autostart entry will automatically start the Centerfield subsystem and server so you do not normally have to start the server manually after an IPL. **If you use the default subsystem (XCSBS80) you can skip the rest of this section.**

The job queue is used as the starting point for starting new HomeRun client jobs. The job queue entry that is added is for the XCTCP job queue that is found in the *{PROGRAM LIBRARY}* library. If you are changing an existing configuration or installing over a previous version of a HomeRun server, the existing job queue entry is first removed and then the new entry is added to the specified subsystem. The command that HomeRun uses to add the job queue entry is similar to the following. The default subsystem is XCSBS80, and the sequence number begins at 50. If 50 is not available, the number is automatically incremented by the HomeRun configuration support until an unused sequence number is found.

ADDJOBQE SBSD(subsystem description**) JOBQ(*{PROGRAM LIBRARY}/XCTCP*) MAXACT(*NOMAX) SEQNBR(50)**

A secondary job queue entry, XCAUTODBA, is used to schedule background work for HomeRun's AutoDBA feature. By default it is placed at SEQNBR(60).

The routing entry that HomeRun installs controls the routing of the HomeRun client jobs. It must be added to the same subsystem that has the XCTCP job queue entry. The routing entry uses the XCTCP routing data for comparison and calls the QCMD program. The XCTCP job class is specified as the job class for the routing entry. The command that HomeRun uses to add the routing entry is similar to the following. The default subsystem is XCSBS80, and the sequence number begins at 170. If 170 is not available, the number is automatically incremented by the HomeRun configuration support until an unused sequence number is found. The second routing entry is added for work that should be run at a lower priority level and therefore uses a different class than the rest of the server jobs.

ADDRTGE SBSD(subsystem description**) SEQNBR(170) CMPVAL(*varies by release*) PGM(QSYS/QCMD) CLS(*{PROGRAM LIBRARY}/XCTCP*)**

ADDRTGE SBSD(subsystem description**) SEQNBR(180) CMPVAL(*varies by release*) PGM(QSYS/QCMD) CLS(*{PROGRAM LIBRARY}/XCTCPBJ*)**

If you are changing an existing configuration or installing over a previous version of HomeRun, the existing routing entry is not removed. It should not cause any harm to your existing application environment to leave it installed in a subsystem that is no longer used by HomeRun. If you want to remove it, you need to remove it using standard work management support available on the IBM System i™. You can use a command similar to following. Before issuing the command, you need to make sure that the sequence number used on the remove routing entry command is the sequence number for the HomeRun routing entry. You can check the routing entries by using the WRKSBSD command and displaying the routing entries for the subsystem that you want to change. For more information about performing work management activities, see IBM System i™ work management documentation.

RMVRTGE SBSD(**subsystem description**) SEQNBR(170)

RMVRTGE SBSD(**subsystem description**) SEQNBR(180)

The HomeRun installation creates objects within the */PROGRAM LIBRARY* installation library to support these and other work management changes on your system. The following are objects that are created by the HomeRun installation support that can be used for work management control.

Object Name	Object Type	Description
XCTCP	Job description	Job description used to set job properties for HomeRun administration client jobs
XCTCP	Job class	Job class used to set job properties for HomeRun administration client jobs
XCTCPBJ	Job class	Jobs class used to run jobs that may run long- running operations and should execute at a lower priority than other work.
XCTCP	Job queue	Job queue used to start HomeRun administration client jobs
XCHLP	Job description	Used by the XCHELPER background job
XCIAFINDER	Job description	Used by the XCIAFINDER background job (V5R4 and higher)
XCPC	Job description	Used by the XCPC background job
XCSTRSVR	Job description	Used by the autostart job entry
XCSCRUBBER	Job description	Job description used to set job properties for the HomeRun Collection Scrubber job

<i>{PROGRAM LIBRARY}</i>	Output queue	Output queue used for upcoming product enhancements
--------------------------	--------------	---

You can modify the objects used for controlling work management, but your changes will not be preserved when you upgrade to the next version of HomeRun.

7.2.5 Exit points

The following exit point is added during HomeRun product installation:

Exit Point	HomeRun Program Name
QIBM_QWT_JOBNOTIFY	XCDTAQ

This exit point is used by the Usage Tracker for insure/INDEX and insure/ANALYSIS. At installation time, no jobs are ended and no subsystems are ended. However, if you want to have the Usage Tracker monitor for specific new jobs that start (that is, if you choose the *Filtered jobs* Profile type), you will need to end and restart any subsystems in which those jobs might start before your monitor will take affect. This only needs to be done once after the installation of HomeRun.

Three other exit points are optionally added at installation time. A prompt will appear that asks if these exit points should be added so that additional types of data collection can be done by HomeRun. By default they are installed, but they can be bypassed if so desired. The three exit points are:

Exit Point	HomeRun Program Name
QIBM_QSQ_CLI_CONNECT	XCCLIINIT
QIBM_QZDA_INIT	XCODBCINIT
DDM exit point (DSPNETA)	XCDDM

7.2.6 TCP/IP Usage

The HomeRun server uses TCP/IP to communicate with the HomeRun clients. The server and client applications communicate almost exclusively using a TCP application. The application is written to use proprietary application data flows to give fast and efficient performance.

The IBM System i™ and personal computer sockets applications communicate with each other through TCP ports. Ports are used by the TCP protocol to identify a unique origin or destination of communication with a TCP application. TCP ports can be any numeric value from 1 to 65535. TCP applications that are commonly used, like ftp and telnet, use pre-assigned port numbers. Pre-assigned port numbers are called well-known TCP ports.

The well-known TCP port numbers range between 1 and 1023 and should not be used when you configure HomeRun. If you specify one of these ports, it can affect the operation of the application that normally uses the well-known port. When you install HomeRun, a port and associated application service is configured. The service name for HomeRun is CENTERFIELD_SERVER_80 and the port by default is **{default port}**. If the **{default port}** port is already used by another application, the port number is incremented until a free port is found. The command that HomeRun uses to add the service and port configuration is similar to the following. You can check the port and service configuration after you install by using the WRKSRVTBLE command and locating the CENTERFIELD_SERVER_80 in the service list.

```
ADDSRVTBLE SERVICE(CENTERFIELD_SERVER_80) PORT({default port})
PROTOCOL('tcp') TEXT('Centerfield Technology Server')
```

In addition to the proprietary communications method, HomeRun uses an ODBC connection to handle report requests made by the administration client. HomeRun requires an iSeries Navigator ODBC connection. When the HomeRun client is installed, the auto-configuration support will attempt to configure an ODBC data source if the supported ODBC driver can be detected.

7.2.7 Job Scheduler Usage

The IBM System i™ built-in job scheduler is used by several components of HomeRun. Events that can cause jobs to be placed on the job scheduler are:

- Scheduling a Database Profiler data collection
- Scheduling Index Create requests using insure/INDEX or Visual SQL Explain
- Cleaning database collections using the Collection Scrubber
- Autonomic Database Assistant (AutoDBA) analysis and actions

7.2.8 Default public authority of libraries

The *{PROGRAM LIBRARY}* and *{DATA LIBRARY}* libraries are created with the default create authority set to *CHANGE.

7.2.9 Command Language (CL) Commands Used by HomeRun

HomeRun is continually being enhanced and modified, so the list of CL commands used by HomeRun also continues to change. The following is a list of CL commands that are used by HomeRun. If you have modified command defaults or installed an application that either replaces or changes any of the following commands, you should contact Centerfield Technology to discuss the impacts that the changes may have on HomeRun.

NOTE: This list may change without notice and is not guaranteed to be complete for the most current version of software.

ADDPFTRG	CHKTAP	DLTJRN	IF	RTVMBRD
ADDMON*	CLOF	DLTJRNRNCV	GOTO	RTVMSG
ADDEXITPGM	CPROBJ	DLTOVR	MONMSG	RTVNETA
ADDJOBQE	CPYF	DLTPGM	MOVOBJ	RTVOBJD
ADDJOBCDE	CPYSPLF	DLTSP*	OPNDBF	RTVSYSVAL
ADDMSGD	CRTDTAARA	DLTSPLF	OVRDBF	SAVOBJ
ADDRTGE	CRTDUPOBJ	DLTUSRSPC	OVRPRTF	SBMJOB
ADDSRVTBLE	CRTJRN	DO	PASSWORD*	SNDPGMMMSG
ALCOBJ	CRTJRNRNCV	DSPDBR	PGM	SNDRCVF
CALL	CRTLTF	DSPFD	PRTSQLINF	STRDBG
CALLPRC	CRTLIB	DSPJOB	RETURN	STRDBMON
CFGSVR*	CRTOUTQ	DSPIOBLOG	RCVF	STRHOSTSVR
CHGACGCDE	CRTPF	ENDDBG	RCVMSG	STRJRNPF
CHGJOB	CRTSAVF	ENDDBMON	RMVJOBQE	STRSVR*
CHGLIBL	CRTSP*	ENDDO	RMVJOBCDE	STRTCPSSVR
CHGNETA	CRTSRCPF	ENDHOSTSVR	RMVMSGD	
CHGOBJOWN	CRTUSRPRF	ENDJOB	RMVPFTRG	
CHGQRYA	DCL	ENDJRNPF	RMVMON*	
CHGSYSLBL	DCLF	ENDPGM	RNMOBJ	
CHGSYSVAL	DLCOBJ	ENDPJ	RSTOBJ	
CHGVAR	DLTDTAARA	ENDSVR*	RTVDTAARA	
CHKOBJ	DLTF	ENDTCPSSVR	RTVJOBA	

* Indicates Centerfield Technology command

7.2.10 OS/400 System APIs Used by HomeRun

HomeRun is continually being enhanced and modified so the list of system application interfaces (APIs) used by HomeRun also continues to change. The following is a list of system APIs and header files that contain API interfaces that are used by HomeRun. If you have installed programs that either replace or change any of the following APIs, you should contact Centerfield Technology to discuss the impacts that the changes may have on HomeRun.

NOTE: This list may change without notice and is not guaranteed to be complete for the most current version of software. This list may not include every API that is used if the API name does not match the included source member (i.e. the name of the API is in the source file and used by Centerfield but not explicitly listed here).

CEELOCT	QUSDLTUS	QWTSETP
QCMDEXC	QUSGEN	QWCRSVAL
QDBRTVFD	QUSLJOB	QUSRTEVI

QDBLDBR	QUSLMBR	QUSMBRD
QLICHGLL	QUSLOBJ	QUSRTVFD
QMHTRTVM	QUSPTRUS	QUSCHGPA
QMHSNDPM	QUSRJOBI	QDBBRCDS
QSYGETPH	QUSRMBRD	QTNADDCR
QSYRLSPH	QUSRTVUS	QUSMIAPI
QUSCHGUS	QWCRJBST	QDMLOPNF
QTNRMVCR	QWCRSSTS	QMHRSNEM
QUSEC	QWCLOBJL	QPMWKCOL
QJOURNAL	QMHCRCVPM	QWDLSJBQ
QTOCNETSTS	QPMLPFRD	QWCLSCDE
QWCCHGTON	QWDLSBSE	
QPMLPMGT	QUSCUSAT	

7.2.11 Programs Adopting *OWNER Authority

The following programs adopt *OWNER authority. All HomeRun programs are compiled to use adopted authority.

Program Name	Description
XCACTODBC	Retrieve actively connected user access jobs
XCADDJE	Add job notify exit point
XCADDJOBCFG	Add job configuration command processor
XCADDUAE	Add the remote monitoring support
XCADVIX	Advise indexes
XCAUDOPR	Audit a client request
XCJBCFGPOP	Change job history prompt override processor
XCCFGMONHS	Configure monitor history auditing
XCCFGMONHP	Configure monitor history prompt override program
XCCHGODBCA	Change run time attributes of a user access job
XCCHKCMTCTL	Check commitment control driver program
XCCMD	Internally used by client software to run a Command Language (CL) command
XCCMT (service program)	Commitment control API interfaces
XCDSPLICINF	Display system license information
XCENDHLPCP	End helper command processing program
XCENDSCRBR	End scrubber job command processor
XCHELPER	Background job reading job notify data queue
XCHLP	Main helper job program

XCIAFINDER	Read journal to retrieve recommended indexes
XCIP	Capture IP address for job
XCJOBUTIL (service program)	Job APIs that require job control authority
XCJDELOG	Look at JDE logs
XCJDEINI	Edit JDE.INI file
XCJOBPRF	Start a database monitor collection for a job
XCLOCKCF	Lock APIs
XCLSTJOB	List jobs APIs
XCMONHST	Monitor job history to audit table
XCMTXWAIT	Mutex APIs
XCPASS	License entry and verification
XCPERFCOLL	Performance collector APIs
XCPMSUB	Dynamically substitute parameters for a parameterized SQL SELECT statement
XCPRFCRT	Create a database collection profile
XCPRFDLT	Delete a database collection profile
XCPRFEND	End a Database Profiler collection
XCPRFENDAP	End all active Database Profiler collections
XCPRFRNM	Rename a database collection profile
XCPRFSTR	Start a Database Profiler collection
XCOWCFG	Configure server for OneWorld environment
XCREPBLD	Build Database Profiler repository
XCREPDLT	Delete repository
XCRTVCFGVAL	Retrieve configuration value
XCRMVJBCFG	Remove job configuration
XCRMVUAE	Remove the remote monitoring support
XCSCDTRG	Schedule requested work (used as trigger program for XCSCD file)
XCSCRUB	Remove unneeded collection data
XCSEMWAIT	Semaphore APIs
XCSIGNON	Check password at sign on time and swap user profile to signed in user
XCSPDSPCMD	Internally used by client software to run an OS/400 command and return spool file output
XCSQLEXEC	Internally used by client software to run non-SELECT SQL statement with commitment control
XCSQLEXEC0	Internally used by client software to run non-SELECT SQL statement without commitment control

XCSQLSEL	Internally used by client software to run SQL SELECT statement with system naming
XCSQLSEL2	Internally used by client software to run SQL SELECT statement with SQL naming
XCSTRHLP	Start helper command processing program
XCSTRSVR	Start the Centerfield server
XCSYSUTIL (service program)	System information APIs
XCTRIGGER	Trigger program for configuration files
XCTRIGGTERM	Trigger program for job monitor configuration files
XCDBGUTIL (service program)	Debug utilities

7.3 HomeRun Tuning Methodology FAQ

The HomeRun toolset employs a tuning methodology that allows you to maximize the overall performance of your database by helping you focus your time and energy on the data access activities that matter most on your system. Whether you are planning for data warehousing projects, debugging existing performance problems, or just trying to get the most out of your system, the methodology employed by HomeRun can help you.

This FAQ employs a whole-toolset approach to SQL tuning. That is, there are suggestions for using each of the tools in the HomeRun toolset. If the FAQ discusses a tool you do not have a license for or are not familiar with, you can find out more from the tool's manual, included on your product CD.

7.3.1 Why tune your database and queries?

If your end users can gather more critical information faster than your competitors can with little or no additional cost to your company, you have a competitive advantage. Business intelligence and data warehousing initiatives are taking companies into an era where, if done right, business analysts can make faster and more precise decisions. Tuning access to your data can give you an additional advantage if you have already deployed a data warehousing solution, or help level the playing field if you have not.

7.3.2 What types of data access tuning can you do?

Data access tuning really involves two types of work, system-wide and individual query tuning. Individual query tuning is done by analyzing each implementation step of a single query and only taking action when you can identify a step that you can improve. On the System i, this has typically been done by running a query under debug and interpreting the query optimizer debug messages in the System i job log or by using output created by the System i database monitor.

System-wide query tuning is tuning by:

- Sampling all query activity on the system
- Ranking the activity on the system by some criteria (typically, either impact to the system or using end user response time)
- Individually tuning highly ranked activities
- Repeating the process

Historically on the System i, we have done virtually all tuning using the individual query tuning methods. Why? It is easy to identify the single long running queries because our users complain about them. Another reason we have always individually tuned our queries is because the tools that are available for doing system-wide query tuning have either been unavailable or have been difficult to use. While system-wide tuning is seldom used, it provides the most benefit to your environment, if the tools are available to facilitate the analysis.

7.3.3 How do you know what to tune?

The most effective tuning requires tools that help you do both system-wide and individual query tuning. The tools should help you identify what data access activities need to be focused on, and either provide you with insight as to what should be done to improve the access, or take some predefined actions to improve it for you.

At a minimum, the tools used for tuning should help you identify:

- What queries provide the longest delays for your end users
- What queries have the largest impact on your system
- What data is accessed most often within your system and when
- Which users or groups of users are accessing your data most frequently
- How are users are getting the information that they want to see

Given these pieces of information, you can focus on the queries, objects, and users that will give the most return for the changes that you make.

7.3.4 What types of things can I do to improve my data access performance?

The actions that you take to improve your data access performance vary based on:

- The transactional uses of your data that is being queried
- The tools being used to run queries on your system
- The level of control you have over the queries being run on your system
- The priority that query activity should have on your system
- The amount of disk resource you have available

On the System i, the activities that have traditionally had largest positive effects on data access performance are:

- Creating a new index that provides a needed access path for the System i Query Optimizer to implement a query in the most optimal way
- Deleting one or more existing indexes that prevent the System i Query Optimizer from examining better access paths and selecting the most optimal implementation of a query
- Changing the query to a form that allows the System i Query Optimizer to better determine the most optimal implementation of a query
- Making program compilation changes to influence the System i Query Optimizer's implementation of a query
- Making program or system changes to influence the operating system's caching and parallel execution algorithms
- Creating a periodically refreshed summary table to simplify and optimize access to the same information
- Preventing estimated long running queries from being run

7.3.5 Specifically, how does HomeRun help me?

HomeRun on the System i gives you the ability to develop, deploy, and manage query activity on your System i with far less effort from you than any methods used in the past. The product has many features that can be used for several purposes depending on your need.

7.3.5.1 For the Programmer

SQL PROGRAM DESIGN AND IMPLEMENTATION:

For programmers writing SQL programs to generate and retrieve new data on the System i, the programmer will start by designing the layout of their program and possibly new database files. While the programmer is doing their design work, they should consider how the System i Query Optimizer will implement their queries. In a best case scenario, the programmer should experiment with the Visual SQL Explain feature to understand how each query will be implemented and understand its impact before finalizing the program design. The information that the feature provides can give valuable insight about needed indexes or may influence some design choices that otherwise may not have been considered.

SYSTEM-WIDE TESTING, DEPLOYMENT, AND PERFORMANCE TUNING:

Once the programmer has completed the design and implementation of the project, system-wide tuning should be performed with either mock system loads or early in the deployment phase. To use HomeRun for system-wide tuning:

- 1) You start by using the usage tracking feature of insure/INDEX or insure/ANALYSIS to sample representative time periods of activity on your system. The usage tracking feature allows you to sample activity either ad hoc or on a scheduled basis. If you choose to track on a scheduled basis, you can sample at specific times of the day, month, or year.
- 2) After you have gathered enough data to be representative of a normal workload, you should run insure/INDEX to identify and create recommended indexes. As you gather more data the indexes recommendations and their associated weights become statistically more accurate. Before acting on the index recommendations, you should consider the impact that your changes will have on your transactional workload. insure/INDEX records which indexes were created using the tool, for easy removal or integration into a source control system.
- 3) You should perform more sampling into a new profile after each set of changes to understand the impact that your previous changes had and find new improvements that can be made.
- 4) After you have created the most beneficial indexes and understand their impact, you can use the reports within insure/ANALYSIS to:
 - a) Identify queries that have the longest delays for your end users
 - b) Identify queries that have the greatest impact on your system
 - c) Identify the most commonly run queries ranked by their cumulative impact

-
- d) Identify queries that may have selected a poor implementation because the System i Query Optimizer timed out choosing an implementation
 - e) Identify the users generating the most query activity
- 5) The reports within insure/ANALYSIS can take you down several paths. You may choose to analyze individual SQL statements, using Visual SQL Explain to fine-tune specific high impact statements. You may choose to create summary tables for commonly run queries. You may use the Database Explorer feature to help you understand why specific queries are timing out. You may choose to use insure/RESOURCES policies to limit the impact of certain users or groups of users.

7.3.5.2 For the SQL Query Designer

QUERY DESIGN AND IMPLEMENTATION:

Query designers writing new queries to retrieve data on the System i will start by understanding the database and the data they intend to access. Typically, the query designer would use the DSPFD, DSPFFD, DSPDBR operating system commands to understand characteristics and relationships within their database. Database Explorer will help the query designer by showing programmer-specified creation detail associated with the different components of the database as well as relationships between the files and indexes within the database. Database Explorer allows the designer to see index search order, data types of fields, and existing logical files and SQL views using a quick and easy, point and click interface. The designer will use the information from Database Explorer to assist coding their SQL in the Visual SQL Explain editor or other favorite tool.

Visual SQL Explain provides all the functionality the query designer needs to design optimized SQL queries. Visual SQL Explain can import SQL query information from the most common applications that query designers may use like Query/400, Query Manager/400, source files, or simply from a PC text file. It will help the query designer understand performance characteristics of an individual SELECT statement by displaying the System i query optimizer's implementation of the statement in a graphical, easy to understand way. The traditional query optimizer debug messages that query designers have used in the past are also provided for those power designers that like the traditional methods. The feature is designed to allow the query designer to quickly iterate through the query design process while continually providing easy to understand implementation feedback. As changes are made to the query, the immediate implementation feedback allows the designer to proactively see how their changes affect the query optimizer's implementation. In the past, the query designer needed to analyze the query's implementation as a separate step that would often be forgotten or bypassed until end users complained about performance.

Visual SQL Explain surfaces the System i settings that affect the query optimizer's choices. The settings are easy to manipulate allowing the designer to quickly experiment with different settings, again using a quick and easy, point and click interface. By

experimenting with the available settings, the query designer can learn how the System i settings affect a query in their environment. It allows the query designer to ensure that the settings are optimal for each query and each environment.

Many times the query optimizer will choose to build one or more temporary indexes to implement a query. If the query requires a temporary index, the time required to build the index is a majority of the overall time needed to run the query. By creating a permanent index to replace an index that the query optimizer chose to build temporarily, the time that was required to build the temporary index when the query was run can almost always be eliminated. Visual SQL Explain graphically surfaces the temporary indexes that the query optimizer believes are needed. If the query designer determines that temporary indexes that query optimizer needs should be created permanently, the wizard for creating indexes makes creating the index a simple process. The wizard starts the query designer in the right direction by defaulting the creation information using information gathered from the query optimizer. It also allows the create index request to be done interactively, submitted for batch processing immediately, or scheduled at a specific time. The index create options give the query designer the flexibility to create the index with as small of an impact on the system as is appropriate for the environment and time of day.

QUERY ONGOING MAINTENANCE:

Once the query designer has designed and tuned the query, insure/RESOURCES can help the query designer understand the query usage and can be valuable for trouble-shooting any problems.