HP Network Node Manager

Reporting and Data Analysis
with
Network Node Manager

HP-UX, Solaris, Linux, and Windows operating systems

Manufacturing Part Number : n/a

July, 2004

1. Introduction
Audience 15
Organization of This Reference 16
The Tasks in this Manual 18
What Is the Data Warehouse? 19
Relational Database Capabilities 20
Searching the Database 21
Generating Reports 21
Extracting Data 22
Adding/Modifying Data 22
Database Administration Tasks 22
Open Database Connectivity (ODBC) 23
SQL 24
Databases in the NNM Environment 25
NNM Internal Databases 25
Database-Related Processes in the HP OpenView Environment 26
Data Warehouse Tools 29

2. Data Warehouse Use Models
Data Warehouse Functionality 34
Data Warehouse Features 35
Distributing the Data Warehouse 36
Disadvantages 36
Using a Specialized Database Server 36
Using a Centralized Database Server 37
Distributing Reporting 38
Using the Data Warehouse 40
Scheduling Data Export 40
Solving Real Problems with the Data Warehouse 41

3. Installation and Configuration
Supported Relational Databases 45
Universal Pathnames and System Independence 45
Choosing a Relational Database 45
Installation and Configuration Process Overview 47
Installing and Configuring an External Relational Database 49

Installing and Configuring an Oracle Relational Database (on UNIX Systems) 50
Installing Oracle 51
Configuring Oracle with ovdbsetup 52
Configuring Oracle Net8. 61
Starting and Stopping Oracle 62
Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems) 64
Configuring Oracle 64
Removing the NNM Database Objects 66
Installing and Configuring Microsoft SQL Server 68
Configuring Microsoft SQL Server. 68
Validating NNM's Connection to a Relational Database 71
Calculating Disk Space Requirements for Trend Data Storage 72
For an Oracle Relational Database 72
For NNM's Embedded Database 72
Considerations for Calculating Disk Space Requirements. 73
Disabling and Enabling the Embedded Database and Web Reporting Interface. 77

4. Migrating NNM Data Between Databases
Migration Considerations 81
Migrating Oracle Data to NNM 6.x 83
Topology Data Migration 83
Trend Data Migration. 83
Event Data Migration. 84
Migrating Data Between Database Products 85

5. Web Reporting Interface
Configuring Reports 89
Using the Report Configuration Interface. 89
Using the Predefined Templates 91
Steps to Configure a Report 92
Viewing Reports 94
Suspending a Report 95
Modifying a Report. 96
Stopping a Report. 97
Deleting a Report 98

Using the GUI 98
Using the Command Line. 98

6. Export Utilities
Explicitly Exporting Data to the Data Warehouse. 103
Managing Topology, Trend, and Event Data 105
Managing Topology Data 107
Managing Trend Data 108
Storing Textual Data 109
Exporting SNMP Trend Data to a Remote Relational Database Server. 110
Managing Event Data 112
Managing Event Filters Using ovdweventflt 113
Using a Scheduler 117
Scheduling Your Trend Exports to Coexist with the Web Reporting Interface. . . . 117
Explicitly Scheduling Export with cron (for UNIX Systems). 118

7. Custom SQL Reporting
Using Schemas 122
Performing Queries on the Data Warehouse 124
Querying the Data Warehouse. 125
SQL Queries of the Topology Schema 125
SQL Queries of the Event Schema. 126
Queries of the Trend Schema. 127
Using SQL Statements for Reporting. 129

8. Accessing the Data Warehouse Schemas
Introduction 133
Entity Relationship Diagrams. 135
General Conventions 139
Language Conventions 140
Topology Data Schema. 141
NNM Objects Table (nnm_objects). 142
NNM Classes Table (nnm_classes) 145
NNM Interfaces Table (nnm_interfaces). 146
NNM Networks Table (nnm_networks) 150
NNM Nodes (nnm_nodes) 151

NNM Segments Table (nnm_segments) 153
NNM Stations Table (nnm_stations) 156
NNM Topology Table (nnm_topology) 159
NNM Nodes Capabilities Table (nnm_nodes_cap) 160
NNM Nodes Network Table 162
SNMP Collected Data (Trend Data) Schema 164
SNMP Trend Version Table (snmp_trend_version) 165
SNMP Trend Description Table (snmp_trend_desc). 166
SNMP Trend Dataset Table (snmp_trend_dataset) 167
SNMP Raw Trend Data Table (snmp_raw_trend) 169
SNMP Reduced Trend Data Table (snmp_reduced_trend). 170
SNMP Aggregated Daily Trend Data Table (snmp_daily_trend). 172
SNMP Aggregated Weekly Trend Data Table (snmp_weekly_trend) 174
SNMP Aggregated Monthly Trend Data Table (snmp_monthly_trend) 176
SNMP Text Collections Data Table 178
Event Data Schema 179
NNM Event Category Table (nnm_event_cat) 180
NNM Event Severity Table (nnm_event_sev) 180
NNM Event Details Table (nnm_event_detail) 182
NNM Event Varbinds Table (nnm_event_varbinds) 184
NNM Event Threshold Table (nnm_event_thresh) 185
NNM Event Aggregation Tables 187
NNM Event Limits Table (nnm_event_limits) 188
Common Data Schema. 189

9. Backup and Recovery
Backup and Recovery. 193
Default solid.ini Scheduled Backup 195
Correcting a Failed Backup 195
NNM Backup Using ovbackup.ovpl 196
Restoring Backups 197
Backup of an Oracle Relational Database 199
Backup of a Microsoft SQL Server Database 200
Independent Backup 201

A. NNM Data Warehouse Troubleshooting

Identifying the Problem Using ovdbdebug (UNIX Only) 205
Using ovdbcheck 207
Examining log Files for Problems 208
Troubleshooting Strategies for Trend Data Storage/Access Problems 209
  How to Recover from ovdwtrend Errors 209
  Avoiding Duplicate and Overlapping Data Records 209
Troubleshooting Database Connection Problems 210
Troubleshooting for the Embedded Database 213
  Problems Starting Database with ovstart ovdbcheck 213
  Repairing a Corrupt Database 213
Troubleshooting for the Oracle Database 216
  Connection Problem 216
  Configuration Problem 216
Troubleshooting for the Microsoft SQL Server 218
  Connection Problem 218
  Data Insert Problem 218
Troubleshooting the Web Reporting Interface 220
  Missing Reports 220
  Performance reports show up in the Report Presenter but contain a message that there is no data on which to report 221
  Threshold Violation reports continuously come up empty 221
  Reviewing Report Schedules Using request_list 222

B. Maintaining a Database
Database Tuning 227
  Improving Performance 227
  Creating Indexes 228
Handling Log Files 229
  For Oracle Relational Databases 229
  For Microsoft SQL Databases 229
Starting and Stopping the Database 231
  Embedded Database 231
  Oracle Database 231
  Microsoft SQL Server Database 231
Authorizing User Access to the Database 232
  Oracle Database 233

Changing the Default User and Password for the Data Warehouse 234
Improving Trend Data Performance 235

C. NNM Data Warehouse and Remote Database Connectivity

SQL Server 2000 as a Remote Database 239
On the Windows machine to be the database server 239
On the Windows machine on which NNM will reside (database client): 239
Oracle 8.1.7 on Windows as a Remote Database 244
On the Windows machine to be the database server: 244
On the Windows machine on which NNM will reside (database client): 244
Oracle 9.2.0.1 on Windows as a Remote Database 250
On the Windows machine to be the database server: 250
On the Windows machine on which NNM will reside (database client): 250
Oracle on UNIX as a Remote Database 256
On the UNIX machine to be the database server 256
Remote Data Warehouse configuration for NNM and Oracle residing on UNIX systems of different types 260

D. Reference Pages

Viewing and Printing Manpages 270

Index 273

Support

Please visit the HP OpenView web site at:

`http://openview.hp.com/`

There you will find contact information and details about the products, services, and support that HP OpenView offers.

You can go directly to the HP OpenView support web site at:

`http://support.openview.hp.com/`

The support site includes:

- Downloadable documentation
- Troubleshooting information
- Patches and updates
- Problem reporting
- Training information
- Support program information

1 Introduction

This chapter introduces the NNM data warehouse which is an integral part of HP OpenView Network Node Manager, both Starter Edition and Advanced Edition. Hereafter, this document simply refers to “NNM”. This chapter describes:

- The audience for this manual.
- The organization of this manual.
- A description of the data warehouse.
- A brief summary of relational database capabilities for database end-users, network administrators, and application developers.
- Definitions for key concepts that illustrate how relational databases work with the HP OpenView environment.

Audience

This manual is written for network administrators, relational database application developers, and network end-users who plan to use the data warehouse to access the NNM data.

If you choose to use an NNM-supported external relational database product instead of the embedded data warehouse relational database, *you must use the vendor's documentation supplied with the external database to install, configure, troubleshoot, and operate the database.* This manual provides some configuration information for external NNM-supported relational database products.

Organization of This Reference

Table 1-1 outlines the contents of this manual. Information that is specific to any of the NNM-supported relational database products is designated by headings identifying the specific database.

Table 1-1 Contents

Chapter 1, “Introduction.”	Describes the data warehouse and its use in handling NNM data.
Chapter 2, “Data Warehouse Use Models.”	Explains how to determine what data to export from the operational databases to the data warehouse.
Chapter 3, “Installation and Configuration.”	Provides procedures for the installation and configuration of the data warehouse and supported relational databases that you can use with the data warehouse.
Chapter 4, “Migrating NNM Data Between Databases.”	Describes how to move data between different NNM-supported databases and different versions of these databases.
Chapter 5, “Web Reporting Interface.”	Describes how to use NNM’s web Reporting interface.
Chapter 6, “Export Utilities.”	Describes the tools provided with the data warehouse that you can use to export data from the operational databases to the data warehouse.
Chapter 7, “Custom SQL Reporting.”	Describes how to query the data warehouse.
Chapter 8, “Accessing the Data Warehouse Schemas.”	Displays the Internet Protocol topology, trend, and event data schema (tables of data) that NNM stores in the data warehouse.
Chapter 9, “Backup and Recovery.”	Describes backup and recovery strategies for the embedded data warehouse, Oracle®, and Microsoft® SQL Server databases.
Appendix A, “NNM Data Warehouse Troubleshooting.”	Describes strategies, including log files and validation tools, for troubleshooting problems that may occur while using the data warehouse.

Table 1-1 Contents (Continued)

Appendix B, “Maintaining a Database.”	Describes tasks a network administrator should routinely perform on a database, such as handling log files.
Appendix C, “NNM Data Warehouse and Remote Database Connectivity,”	Describes tasks to set up the data warehouse on a remote database server.
Appendix D, “Reference Pages.”	Lists the reference command (manpages and reference pages) that are related to topics described in this manual, and provides information on how to access the manpages online.

The Tasks in this Manual

The tasks listed below are in the order that they need to be completed to set up the data warehouse. NNM is assumed to be installed and running.

1. Understand the organization of the NNM data and the data warehouse. For more information, see the remaining sections in Chapter 1 and Chapter 2, “Data Warehouse Use Models.”
2. Choose a relational database. Install and configure your relational database if it is not already done. See Chapter 3, “Installation and Configuration.”
3. Migrate data obtained using a version of NNM prior to 6.0 or a different database to the current version of NNM and your current database. Refer to Chapter 4, “Migrating NNM Data Between Databases,” for more information.
4. Export data from the internal NNM databases to the data warehouse. Refer to Chapter 6, “Export Utilities.”

NOTE At this point, you have completed all the tasks necessary to populate the data warehouse with data. The following procedures can be completed as needed.

5. Trim event and trend data from the data warehouse and internal NNM databases. Refer to Chapter 6, “Export Utilities.”
6. Perform queries on the data warehouse databases using the schemas and guidelines provided in Chapter 7, “Custom SQL Reporting,” and Chapter 8, “Accessing the Data Warehouse Schemas.”
7. Use the NNM web reporting interface to generate reports. Refer to Chapter 5, “Web Reporting Interface.”
8. Complete backups of your relational database on a regular basis as described in Chapter 9, “Backup and Recovery.”

What Is the Data Warehouse?

A data warehouse is a repository of data about a related set of objects that is used to understand how to optimize a business process. NNM contains network data that can provide information about frequency of network problems, usage of network elements, and trends of network performance.

The data warehouse is a repository of NNM trend, event, and topology data stored in a relational database to be used for analytical purposes. The data warehouse can use either NNM's embedded relational database or an HP OpenView-supported external relational database as the repository.

The data warehouse provides tools to export, aggregate, trim, and query the data using Open Database Connectivity (ODBC) technology. The open schema allow SQL (Structured Query Language) access from reporting tools or analysis engines.

To understand how the data warehouse operates, you need to understand the following concepts:

- Relational database capabilities (see “Relational Database Capabilities” on page 20).
- ODBC (Open Database Connectivity) and how it allows access to the database (see “Open Database Connectivity (ODBC)” on page 23).
- SQL (Structured Query Language) and how it is used to access data in a relational database (see “SQL” on page 24).
- NNM databases that contain the network data (see “Databases in the NNM Environment” on page 25).

Relational Database Capabilities

NNM uses a relational database and provides the following database capabilities. If you are using an external NNM-supported database, details on which programs and procedures to use for each capability can be found in the documentation for the specific database you are using.

NOTE See the Release Notes, Supported Configurations topic for information on supported versions.

HP OpenView Network Node Manager supports three relational database products:

- Embedded data warehouse database (for UNIX® or Windows operating systems)

  The embedded database is installed and configured during the installation of NNM. The embedded database is an excellent choice for handling NNM data. The database is robust and grows dynamically as more data is introduced into the database.

  Note that there are few third party reporting tools that can use the embedded database technology.

- Oracle Server (for UNIX or Windows operating systems)

  Some releases of Oracle Server are supported only on specified platforms. Refer to the Release Notes for the NNM product to determine which releases of Oracle Server are supported on specific platforms.

  Oracle is available from Oracle Corporation.

- Microsoft SQL Server (for Windows operating system only.)

  SQL Server is available from Microsoft Corporation.

Common tasks carried out with a relational database are listed below:

- Search the data stored in the database.
- Generate reports with customized formats, charts, and graphics.
- Export data to other databases or text files.

- Add or modify data and tables in the database after obtaining the proper security and access permissions.
- Carry out the database administration tasks.

Searching the Database

A relational database has a unique model and structure that lets you search in the database. Unlike flat file or hierarchical databases, a relational database stores data as values in tables and allows you to search for the data value itself, which might be found in one, several, or all tables in the database. You do not need to navigate directory paths or specify exactly one location for the data.

The interface between the relational database and the NNM platform is SQL (Structured Query Language). SQL enables you to query the database to return data that matches specified criteria. You can use SQL to:

- Compute a specific data result.
- Search for data in a single table or "join" data from several tables.
- Divide a query into several subqueries, each one retrieving a specific data value. The results of the subquery are combined to obtain the "complete picture."

Many relational databases and third-party vendors provide utilities or programs which facilitate the task of querying a database. These utilities include: online "query forms," supplemental search commands in pull-down menus or dialog boxes, and interactive prompts.

The data warehouse provides the `ovdwquery` tool to send SQL commands to a database. The query tool is described in Chapter 7, "Custom SQL Reporting."

Generating Reports

To convert "raw" data into a more usable format, you can use a program or utility that provides preformatted report templates. The report template is used with the database to format the data retrieved by a query. These utilities and programs often have graphic capabilities that enable you to add graphics and charts to your reports.

NNM provides a web Reporting interface that allows you to configure and present reports in HTML format. This is described in Chapter 5, "Web Reporting Interface."

Extracting Data

Extracting data is another task for the database. Data is extracted from the database for the following reasons:

- To transfer data from one database to another database on the same network.
- To transfer data between databases on different networks.
- To output data from a database into a different format, such as online text files or hardcopy reports.

Adding/Modifying Data

The database is dynamic. Data can be added to or deleted from the database. Data can be modified in the database. Some changes to the database require that access permission be obtained from the system administrator.

Database Administration Tasks

System administrators and application developers may also use the following administrative capabilities of the relational database:

- Creating a security mechanism that controls user and administrator access to a database.
- Using proprietary programming languages, constructs, or SQL extensions to create a database or its contents.
- Managing and modifying database schema tables.
- Customizing any forms or report-generation tools supplied with the relational database.
- Integrating third party applications with HP OpenView to provide additional network and systems management functionality.

Open Database Connectivity (ODBC)

ODBC (Open Database Connectivity) is a widely accepted application programming interface for database access. It is based on the Call-Level Interface specifications from X/Open for database APIs and uses SQL as its database access language.

ODBC is designed for maximum interoperability. That is, the ability of a single application to access different DBMS (database management systems) with the same source code. Database applications call functions in the ODBC interface, which are implemented in database-specific modules called drivers. Because drivers are automatically loaded when the database is started, an administrator only has to add a new driver to access a new DBMS; it is not necessary to recompile or relink the application.

The **ODBC data source** is a logical name that links the physical database product to a logical name. The ODBC data source uses a configured driver that allows you to connect to a database.

ODBC is pervasive in Windows operating systems, but is much less common in UNIX environments. NNM supplies the ODBC environment for UNIX for use with the data warehouse.

SQL

SQL (Structured Query Language) is both a language and a tool for communicating with a relational database. SQL is formally recognized as an ANSI/ISO standard, and SQL is the industry-accepted standard language that enables you to create and operate on a relational database. SQL is the language used to communicate with relational databases in the HP OpenView environment.

Some of SQL's major features include:

- For database users and administrators, SQL provides a set of query commands to access data in the database.
- For database administrators, SQL serves as a tool for distributing data across multiple workstations and servers within a network. Each workstation and server uses SQL as a common language for sending, receiving, and sharing data across the databases in the network.
- For database application developers, SQL commands are programming commands that enable a developer's program to access data from a database.

Most commercial database products extend SQL beyond the ANSI definition. These extensions consist of expanded parameter sets or syntax revisions, both of which provide broader capabilities for manipulating the data in a database. For example, Oracle has added categories to the basic SQL statement set that include session control statements, system control statements, and embedded SQL statements. Oracle also provides PL/SQL, a procedural language.

There are numerous books and guides available that describe how to use SQL. If you are using a commercial relational database, your vendor's documentation should include SQL and SQL-extension information.

Databases in the NNM Environment

In an NNM environment, the relational database interacts with several NNM components such as databases, interfaces, processes, and local registration files. These components are briefly described in this section.

The databases are internal databases that are used by the NNM processes. These databases are not directly available through the data warehouse, although most of the information is exported into the data warehouse.

NNM Internal Databases

The NNM environment uses five types of databases for its operations. Each type of database provides storage for a different kind of information. However, your network may not have five physical, stand-alone databases.

The five database types are:

- **OVW object database**

  Manages all object and field information for HP OpenView Windows. This object database is not stored in your relational database.

- **Map database** (at least one)

  Contains presentation information specific to a map. There is one map database for each map in your NNM environment. Examples of presentation information include: symbol labels, the correlations between symbols and their associated specific objects, and the placement of a symbol on the map. Map databases are not stored in your relational database.

- **Topology operational database**

  Contains the basic information regarding nodes and interfaces of the devices on a network. The topology data is an “inventory” of your network. Examples of topology data include node names, IP addresses, and router interface information.

  Much of the information in the topology tables is duplicated in the OVW object database. One important type of information not duplicated in the OVW object database is state information for netmon (the IP discovery and status monitoring process), including time stamps that indicate when the object last changed and when the object should next be polled. This information helps netmon detect changes so netmon can communicate the changes to ovtopmd and to pmd. (The topology tables are controlled by ovtopmd and are updated based on information received from netmon and edits to the map. pmd is an event-forwarding agent.)

You can export this data to the data warehouse. Note that the data warehouse does not retain historical topology information, only a current snapshot.

- **SNMP-Collected binary database**

  Contains SNMP data as collected by snmpCollect. This data is used by xnmgraph and other NNM facilities. The data is stored in a proprietary database format, but can be exported to the data warehouse using ovdwtrend.

- **Event database**

  Contains event data from the postmaster, pmd. This data is used by many NNM facilities, including the Alarm Browser and xnmappmon. The data is stored in a proprietary database format, but can be exported to the data warehouse.

Database-Related Processes in the HP OpenView Environment

The five database types in the NNM environment are controlled by NNM processes. You cannot edit these databases directly.

The foreground and background processes collect network data, store it in the appropriate database locations, and retrieve and format the data in response to queries.

1. Each background process in NNM has its own local registration file (LRF). An LRF is a specially-formatted ASCII file that contains information about a process such as its name, how to start it, dependencies it has on other processes, command-line arguments, and which objects (if any) the process is responsible for.
2. Information in the LRF serves as entries in the ovsuf start-up configuration file. You can modify ovsuf only by using the ovaddobj and ovdelobj commands.

3. The `ovstart` foreground process contacts (and starts, if necessary) the `ovspmd` management process management.
4. `ovspmd` starts running any background processes that were specified in the `ovsuf` start-up configuration file. The dependency information in `ovsuf` tells `ovspmd` in what order to start all the processes.

   NNM has many processes. The background processes specifically related to database operations are: `netmon`, `ovtopmd`, `ovwdb`, `pmd`, `ovrequestd`, and `snmpCollect`. These processes are described below.

NNM Processes Related to Database Operations

`netmon`

`netmon` continuously monitors the state of your network using SNMP requests and ICMP requests to remote systems. `netmon` communicates any changes it finds to `ovtopmd` and sends the necessary corresponding events to `pmd`. `ovtopmd` updates the topology tables and the object database. If the change affects the presentation, `ipmap` (the process that draws and updates network maps) acts on it. The object database is not stored in your relational database.

Likewise, when you edit the map, change `netmon`'s polling values, or when the status of nodes propagates up to the segment or network containing them, `ipmap` communicates these changes to `ovtopmd`, which updates the topology database with the changes and sends corresponding events to `pmd`. Once the changes have been added to the topology database, `ovtopmd` notifies `ipmap`, and the map of your IP network is automatically updated.

For more information about NNM processes, refer to Appendix B in the manual, *Managing Your Network with HP OpenView Network Node Manager*.

`ovrequestd`

This background process schedules the data warehouse and reporting tasks for the web Reporting interface. NNM uses the configuration part of the web Reporting interface to determine what reporting tasks to add and remove from `ovrequestd`.

`ovtopmd`

This background process maintains the network topology database. The topology stores `netmon` polling values and information about network objects, including their relationships and status. `ovtopmd` reads the topology database at start-up and populates this database with network objects during discovery of your network's topology. Once the topology has been generated, `ovtopmd` controls the topology and updates specific records, based on information from `netmon` indicating that changes have occurred in the corresponding nodes.

NOTE It is recommended that the database server and `ovtopmd/netmon` run on the same system. If you do not use this configuration, you cannot use HP OpenView to debug any network problems if the database cannot be accessed.

`ovwdb`	This process controls the OVW object database, which stores information about objects. These objects may be represented on the map by one or more symbols. If `netmon` detects a change in the network, `netmon` calls `ovtopmd`, which calls `ovwdb` to update the OVW object database.
`pmd`	This process multiplexes and logs events. `pmd` logs all events to the files `$OV_DB/eventdb/*`. `pmd` also forwards events from the network on to other applications that have connected to `pmd` using the SNMP API.
`snmpCollect`	This process collects MIB data and performs threshold monitoring. `snmpCollect` stores the data it collects in the `$OV_DB/snmpCollect` directory and sends threshold events to `pmd`. You can use `ovdwtrend` (described in the next section) to export collected data to a relational database.

For more information about NNM processes, refer to Appendix B in the manual, *Managing Your Network with HP OpenView Network Node Manager*.

Data Warehouse Tools

The background processes described in the previous section enable you to carry out routine database work on the operational databases. The data warehouse provides tools to export data from the operational databases into the data warehouse and to create reports using this data. These tools are described below.

General-Purpose Tools

`ovdwquery` takes a query written in SQL, sends the query to the relational database configured as the NNM data warehouse, and returns the results of the query to the standard output.

`ovdbcheck` starts, stops, and monitors the condition of the embedded database server when the embedded database is used. `ovdbcheck` validates that the data warehouse tools are able to connect to a relational database.

`ovdbdebug` validates the connection between a relational database and the `openview` database. If the connection cannot be made, `ovdbdebug` provides error messages for use in resolving the connection failure.

`ovdwconfig.ovpl` configures the data warehouse utilities to use the desired database product, ODBC data source, and specified database authorization.

`ovdbsetup` is an optional tool that Oracle for UNIX database users can use to configure an Oracle database instance for the OpenView database.

Data Collector Tools

`ovdwtrend` performs export, sum, and trim operations to maintain data in a relational database's raw and reduced SNMP trend data tables. HP recommends the use of `ovdwtrend` for all data collector operations. `ovdwtrend` combines the functions of `ovcoltosql` and `ovcoldelsql` into one tool. You can also access the `ovdwtrend` command through the NNM menu item `Tools:Data Warehouse->Export Trend Data`.

`ovcoltosql` reads data collector binary files in `$OV_DB/snmpCollect`, optionally summarizes that data, optionally deletes binary data, then writes either the raw or reduced data to a relational database.

`ovcoldelsql` deletes all or part of the data in the relational database's raw or reduced trend data tables.

`ovcolqsql` performs SQL queries to retrieve and summarize data in a relational database's raw and reduced trend data tables. `ovcolqsql` is maintained for backward compatibility. See `ovdwquery` under General-Purpose Tools.

See also `ovdwquery` under General-Purpose Tools.

Topology Tools

`ovdwtopo` performs topology maintenance for the data warehouse. You can access the `ovdwtopo` command through the NNM menu item `Tools:Data Warehouse->Export Topology`.

See also `ovdwquery` under General-Purpose Tools.

Event Tools

`ovdwevent` performs export and trim operations to maintain data in the data warehouse event tables. You can access the `ovdwevent` command through the NNM menu item `Tools:Data Warehouse->Export Events`.

`ovdweventflt` configures filters to restrict events exported to the data warehouse.

See also `ovdwquery` under General-Purpose Tools

Migration Tools

`ovdwunloader` copies the contents of the data warehouse to an intermediate file (optionally with fields separated by commas).

`ovdwloader` loads the contents of the intermediate file from `ovdwunloader` or `ovcolmigo` to the data warehouse.

`ovcolmigo` copies the contents of NNM 4.1 or NNM 5.0x SNMP trend tables to an intermediate format for loading into NNM 6.x.

Reporting Tools

`nnmRptConfig.exe` allows you to configure reports using the web Reporting interface. You can access the `nnmRptConfig.exe` command using the NNM menu item `Options:Report Configuration`.

`nnmRptPresenter.exe` allows you to view the reports using the web Reporting interface. You can access the `nnmRptPresenter.exe` command using the NNM menu item `Tools:Report Presenter`.

2 Data Warehouse Use Models

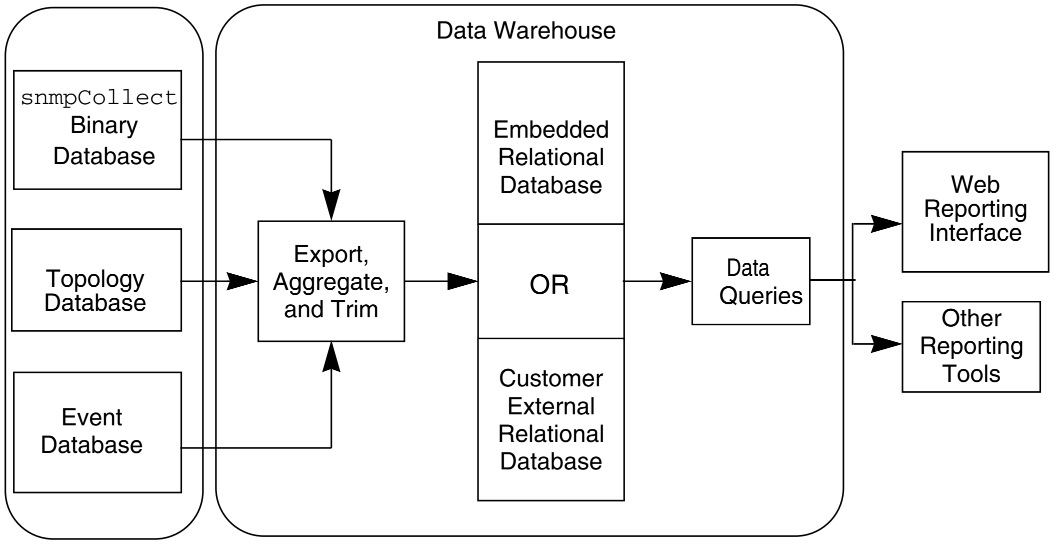
Data Warehouse Functionality

The data warehouse provides the functionality to access the network information available in HP OpenView NNM. This network information may help you to identify problems in your network or show trends of network performance.

A data warehouse is a repository of data used to understand how to optimize a business process. The data warehouse stores the data in a relational database and provides access to that data via SQL-based queries.

The operational databases export data to the data warehouse tables for analysis (see Figure 2-1).

Figure 2-1 Data Warehouse as an Analytical Repository


Data Warehouse
snmpCollect Binary Database
Topology Database
Event Database
Export, Aggregate, and Trim
Embedded Relational Database
OR
Customer External Relational Database
Data Queries
Web Reporting Interface
Other Reporting Tools

Data Warehouse Features

A data warehouse is a collection of information about a related set of objects. The information is usually stored in a relational database. A data warehouse is designed so that users can access the database with a query tool to obtain the information they need.

A data warehouse enables the user to:

- Perform advanced analysis on specific sets of data by using the most appropriate reporting tool.
- Develop reports that meet the specifications of the user.
- Enable a variety of users to analyze data in the data warehouse.

NNM's data warehouse provides tools to analyze NNM data. NNM exports trend, topology, and event data to the data warehouse. The data warehouse provides interfaces for customization, activation, scheduling, and export of the data. For example, you can use operations in the data warehouse to:

- Export trend, topology, and events data from the NNM topology database.
- Aggregate trend and event data into daily, weekly, and monthly tables for long term trend analysis. (Event data is aggregated into daily, weekly, monthly, and yearly tables as part of the export process.)
- Trim trend and events data after a specified period of time.
- Delete the `snmpCollect` data.

Distributing the Data Warehouse

It is often more convenient to centralize your data processing on a system other than your NNM management station. The data warehouse provides two mechanisms for this: using a database server on a node separate from a single NNM management station (specialized server), and centralizing data from multiple collection stations onto a single management station (centralized server).

Disadvantages

While centralizing your database operations may be convenient, it can have some disadvantages.

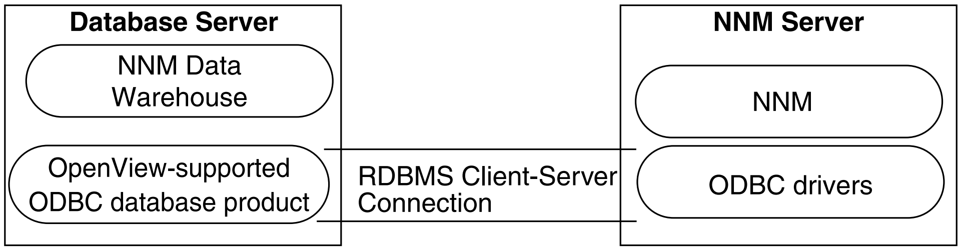
NNM is your primary tool for monitoring and managing your network. If you have a network failure between your management station and your database server, you may be unable to access your stored information.

Also, transferring all of your database information across the network may cause significant traffic between your management station and your database server.

Using a Specialized Database Server

By configuring your commercial database, such as Oracle or MS SQL Server, to automatically use an ODBC link that extends to a database which physically resides on another machine, you can dedicate one system to network management work and another to database work. This is shown in Figure 2-2. Refer to the Remote Database Connectivity whitepaper on your product CD for more information.

Figure 2-2 Specialized Database Server


Database Server
NNM Data Warehouse
OpenView-supported ODBC database product
RDBMS Client-Server Connection
NNM Server
NNM
ODBC drivers

Using this model, your export operations transparently pass the data through the ODBC driver, across the network, and into the data warehouse on the database server system.

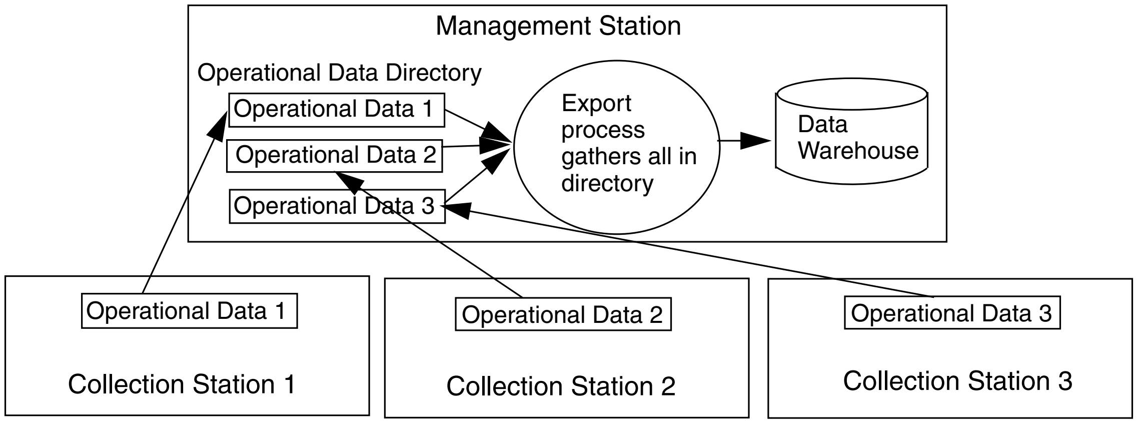
This model is not supported for the embedded database.

NOTE Only one exporting client may access the database on the server using this model. Using multiple clients will probably cause data corruption.

Using a Centralized Database Server

When you want to combine trend (SNMP Collect) data from multiple client network management stations onto a single management station's data warehouse, you must manually upload the operational data prior to executing your export command on the management station. This is shown in Figure 2-3.

Figure 2-3 Centralized Database Server



This model takes advantage of the fact that the export operation collects all the data in the `$OV_DB/snmpCollect` directory and exports it all at one time to the data warehouse. In the case of any overlaps, the first processed value remains in the database.

To collect your operational data into the central directory, use the `ftp` command. Refer to "Exporting SNMP Trend Data to a Remote Relational Database Server" on page 110 for more information.

Distributing Reporting

If you have implemented NNM in a distributed manner, that is, with NNM collection stations reporting to one or more management stations, you have two deployment options for using the NNM Reporting interface:

1. Configure each collection station to generate reports for the nodes it manages. View these reports individually via the web from any system running an NNM-supported browser.
   - Advantages:
     - SNMP polling is distributed, reducing network load and congestion.
     - Data warehouse storage is distributed, thereby distributing disk space consumption.
     - Report generation processing is distributed across multiple computers, shortening report generation time.
     - For most configurations, reports can be generated for a greater number of nodes/interfaces in total.
   - Disadvantages:
     - You do not see a single, unified report for the entire network.
     - You have to perform report configuration separately on each collection station.
2. Configure report generation only on the management station. Again, report viewing is from any system running and NNM-supported Web browser.
   - Advantages:
     - You get a single, unified set of reports for the entire management domain.
     - You only configure report generation once, on the management station.
   - Disadvantages:

- All SNMP polling for the reports originates from the central management station.
- This requires that the management station be configured with the SNMP GET community names of all target nodes. It also may result in high levels of SNMP traffic over slow or expensive links.
- If reports are generated for all or many nodes across the entire management domain, disk storage requirements could be quite larger.
- The total number of nodes/interfaces for which reports can be generated is more restricted than in alternative one.

3. You may use a combination of alternatives one and two. Specifically, use alternative one for some reports and alternative two for others.

Using the Data Warehouse

NNM collects topology, trend, and event data that is used to troubleshoot, report on, and analyze the network. The data warehouse provides both a repository for NNM data and tools for analysis of the data.

Some reports are automatically configured when you install NNM. To see what reports NNM automatically configures, use the `Options:Report Configuration` menu item. From the `Create NNM Reports` interface, double-click on `View Schedule Reports` from the scoping pane.

If you want to discontinue collecting data for the reports, you must stop each report. See “Stopping a Report” on page 97 for information.

Scheduling Data Export

Prior to analysis, the data must be exported to the data warehouse. The purpose of the data warehouse is to facilitate the analysis of network data. When data is used for analysis rather than troubleshooting, the data can be historic. Analysis does not require absolutely current data.

Reporting generated from the web Reporting interface automatically schedules data export. If the pre-designed reports do not fill your needs, you may design your own reports and schedule your own exports.

To simplify the export of data, you can set up a scheduler to automatically export data hourly. `cron` is available on UNIX systems. Microsoft SQL Server provides its own scheduling utility. To get an accurate snapshot of the state of your network, you should schedule data warehouse tools to run sequentially.

NNM daily performance reports for the web Reporting interface require that data reside in the Data Warehouse for at least a full day. If you have configured NNM performance reports and use the `ovdwtrend -trim` command, be sure to specify a `-trimpriorto` value greater than 48 hours. That is, 24 hours for yesterday and up to 24 hours for today.

Solving Real Problems with the Data Warehouse

The analysis and compilation of data from the data warehouse enables you to solve real network problems. Reports that compile the data provide information such as network utilization and event reports. You can generate these reports many times by creating a template in a report generation program.

You can use reports to address a variety of network issues as are described in Table 2-1.

Table 2-1 **Examples of Reporting Using the Data Warehouse**

Report Examples	Benefits
Routine reports for management	Provide information such as network utilization and event reports.
	Can use the web Reporting interface which runs the report many times based on the template.
Analytical reports solving a specific problem	May address a longer term issue, such as a change in the device’s network utilization from year to year.
	Enable users to proactively identify resource issues by creating a baseline and comparing subsequent traffic against the baseline.
Proactive network analysis	Determine thresholds for critical devices on the network.
	Use the data warehouse as a source of historical information to determine baselines by analyzing data in the reports.
	Observe movement in baselines indicating a change in network performance.

3 Installation and Configuration

This chapter describes the installation and configuration of the data warehouse. The configuration information in this chapter is dependent upon your choice of relational database. The data warehouse provides its own embedded database, or it can be configured to use Oracle or Microsoft SQL Server.

NOTE If you plan to use Oracle or Microsoft SQL Server, refer to the database's vendor documentation for specific installation instructions. This chapter provides guidelines, not actual procedures, for the installation of external NNM-supported databases.

Supported Relational Databases

NNM supports an embedded relational database and two external relational database products. Oracle Server is available from Oracle Corporation. Microsoft SQL Server is available from Microsoft Corporation.

NOTE See the Release Notes, Supported Configurations topic for information on supported versions of these databases.

Universal Pathnames and System Independence

NNM supports multiple operating system platforms, with differing file structures. The different arrangements for files on these different systems could complicate procedural instructions.

To simplify your use of the product and make the documentation more readable, the NNM products include scripts that define environment variables common to all supported operating systems. These environment variables create “universal” path and file names that apply to NNM directories and files regardless of the structure of the underlying file system. The procedures described in this manual use, whenever possible, universal pathnames for commands that you execute.

For more information about universal pathnames and instructions to setup universal pathnames, refer to the *ov.envvars* manpage or reference page.

For example, a written procedure might include the following step:

Windows®:	`install_dir\bin\ovdbsetup -o`
UNIX:	`$OV_BIN/ovdbsetup -o`

Choosing a Relational Database

The embedded database was installed and configured when you installed NNM. The embedded database is completely integrated into NNM (including backup, starting and stopping, and configuration). The database requires some customer interaction, for example, deleting or monitoring the size of the database so that it does not fill up your disk. This embedded database technology is found in other HP OpenView products.

The data warehouse also supports Oracle and Microsoft SQL Server relational databases. You must install and configure these databases separately from NNM.

The data warehouse supports several options for handling the NNM data:

- If you plan to use the installed and configured NNM embedded database, go to “Validating NNM’s Connection to a Relational Database” on page 71.
- If you plan to use an external NNM-supported relational database such as Oracle or SQL Server, you must first install and configure the database. See “Installing and Configuring an External Relational Database” on page 49.
- If you do not plan to use the data warehouse, you can disable the embedded database. See “Disabling and Enabling the Embedded Database and Web Reporting Interface” on page 77 for instructions on how to disable and re-enable the embedded database.

NOTE NNM does not require that a database be operational on an NNM management station. Since the data warehouse is not necessary for NNM to run, the database can be set up on a different server. If you use Oracle’s Relational Database or Microsoft SQL Server, see your vendor’s documentation for more information about configuring remote clients.

Installation and Configuration Process Overview

The following table shows the general processes for using NNM with Oracle, whether you are installing a new version or upgrading from a previous installation.

Table 3-1 Installing and Configuring NNM with Oracle

	UNIX	**Windows operating systems**
Clean install (first installation or previous version removed)	Install NNM Install Oracle `ovdbsetup -o`	Install NNM Install Oracle Configure Oracle `ovdwconfig.ovpl`

Upgrade
(previously used Oracle)
	`ovdwunload`
	Install NNM
	`ovdwload`
		`ovdwunload`
		Install NNM
		`ovdwload`

NNM provides two primary tools for use in configuring relational databases, `ovdbsetup` and `ovdwconfig.ovpl`. The following table shows when to use each of the tools.

Table 3-2 Configuration Tool Comparison

	ovdbsetup on UNIX	**ovdwconfig on UNIX**	**ovdwconfig on Windows operating systems**
The destination database must be...	Oracle, installed	any, running	any, running
Create an Oracle database	X		

Table 3-2 Configuration Tool Comparison (Continued)

	ovdbsetup on UNIX	**ovdwconfig on UNIX**	**ovdwconfig on Windows operating systems**
Configure an Oracle database	X		
Switch from... to...	embedded Oracle	any any	any any
Drop and reconfigure NNM tables		X	X
Change database user and/or password		X	X
Change the ODBC datasource environment variable		X	X

NOTE Suppose you purchased NNM Advanced Edition (NNM AE) and are using NNM AE's Extended Topology functionality. If you plan to use Oracle as an external NNM-supported relational database, the stored data from NNM AE's Extended Topology functionality will reside within the Oracle database.

Installing and Configuring an External Relational Database

If you choose to use your own external relational database, not the NNM embedded database, you must install and configure that relational database. Follow the installation and configuration guidelines in the next section for your database. For more specific information, go to your vendor's database documentation.

The database configuration information is the same, whether you are storing topology data, trend data, event data, or any combination of the data types. You need to create an `openview` database to store NNM topology, trend, and event information in the relational database.

The data warehouse databases will need enough disk space to store large amounts of NNM data. To help you determine how much disk space you should allocate, see "Calculating Disk Space Requirements for Trend Data Storage" on page 72.

To install and configure an external NNM-supported relational database, follow the installation and configuration instructions for your relational database:

- "Installing and Configuring an Oracle Relational Database (on UNIX Systems)" on page 50.
- "Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems)" on page 64.
- "Installing and Configuring Microsoft SQL Server" on page 68.

NOTE The default configuration of NNM's data warehouse to an external database assumes a database installed on the same machine as NNM. While this configuration is often adequate, there are circumstances where separating NNM application from the database server is desirable.

See Appendix C, "NNM Data Warehouse and Remote Database Connectivity," on page 237 for instructions on using a database residing on a separate machine.

Installing and Configuring an Oracle Relational Database (on UNIX Systems)

The Oracle database can be installed as a stand-alone workstation server or on a remote server. Refer to your Oracle documentation for information about the Oracle products. The products shown below define the minimal set that must be installed.

You must update the environment variables in the `ovdwenvs.ovpl` file when you upgrade Oracle versions. To configure `$ORACLE_HOME` and other environment variables in the `ovdwenvs.conf` file, use the `env` option with `ovdwconfig.ovpl` command. For example, see the italicized part of the following:

```
/opt/OV/bin/ovdwconfig.ovpl -env "ORACLE_HOME=/opt/oracle/product/9.0.1"\
-type oracle -rdb OVoracle
```

NOTE See the Release Notes, Supported Configurations topic for information on supported versions of these databases.

- Oracle server
- SQL*Plus
- Net8
- TCP/IP protocol adapter for Net8
- The following components are not required by NNM, but can be used by you to develop your own reports:
  - Oracle Data Query
  - SQL*ReportWriter
  - SQL*Report
- If you are configuring Oracle on a remote server, the database client (NNM system) requires:
  - Oracle client
  - Net8

— TCP/IP protocol adapter for Net8

Installing Oracle

To install Oracle products, follow the instructions below and refer to the Oracle documentation for specific procedures:

1. Set the following environment variables in the start-up scripts for users `oracle` and `root`. (Bourne shell and `ksh` users need to modify their `.profile` scripts, while `csh` users need to modify their `.login` script.) The variables to be set and exported are:

   `ORACLE_SID` = **`openview`**

   `ORACLE_TERM` = *`terminal_type_you_are_emulating`*

   `ORACLE_HOME` = *`home_directory_for_this_Oracle_instance`*

2. For user `oracle`, set your path to include `$ORACLE_HOME/bin`. In your home directory, execute the following commands:

   If you are running `sh` or `ksh`, execute:

   ```
   PATH=$PATH:$ORACLE_HOME/bin
   ```

   If you are running `csh`, execute:

   ```
   set path=($path $ORACLE_HOME/bin)
   ```

   The data warehouse commands, `ovdbsetup` and `ovdwconfig.ovpl`, also require that these environment variables be set. You may want to modify your root user `.profile` or `.login` scripts to set these variables.

3. Check the Oracle installation prerequisites. (If you are also planning to use the Oracle Relational Database with an HP OpenView product other than NNM, be sure to check that product's documentation for information on relational database prerequisites.)

4. As `root`, perform the necessary pre-installation tasks:

   - Create an Oracle Relational Database owner (`oracle`).
   - Create directories as required by Oracle.
   - Modify operating system parameters to satisfy Oracle requirements.

5. As the Oracle owner, install the relational database product with the Oracle Installer program, `runInstaller`.

NOTE

If you plan to use the NNM database creation program, `ovdbsetup`, to create the Oracle instance to be used by the data warehouse, select the “Install/Upgrade Software Only” option when prompted by `orainst`. This option installs the database server without creating a database.

6. As `root`, run the Oracle post-installation script as directed by Oracle. Enter:

   ```
   $ORACLE_HOME/orainst/root.sh
   ```

NOTE

You can use the data warehouse with existing Oracle instances that may be shared with other applications. To configure Oracle to run with the data warehouse and other applications, run `ovdbsetup` as described below in “Configuring Oracle with ovdbsetup” on page 52.

NOTE

If you have questions or problems while installing and configuring the Oracle Relational Database, contact your Oracle software vendor directly for help.

Configuring Oracle with ovdbsetup

To configure the Oracle database after completing the installation, follow the guidelines below:

1. Log in as user `root`.
2. Set the `$ORACLE_HOME` environment variable for your default login script. (Bourne shell and `ksh` users need to modify their `.profile` scripts, while `csh` users need to modify their `.login` script.)

   `ovdbsetup` may prompt you to set up the `$ORACLE_BASE` and `$ORACLE_SID` variables as well, if they have not yet been defined.
3. Initialize the NNM environment variables. Type:

   ```
   /opt/OV/bin/ov.envvars.sh
   ```

4. Run the database setup script, `ovdbsetup`. Type:

   `$OV_BIN/ovdbsetup -o`

   The script prompts you for configuration information.

   NNM's `$OV_DB` is under `/var/opt/OV/`. In keeping with this convention, you may want to use `/var` for data and index directories (for example, `/var/oradata/openview`).

CAUTION Ensure that the paths used for the data and index directories have sufficient disk space for your needs before proceeding. Insufficient disk space causes ovdbsetup to fail.

Determine which file systems have at least 350M of available space. For information on the amount of disk space required, see "Considerations for Calculating Disk Space Requirements" on page 73.

To verify the amount of space available, type `bdf -l` on HP-UX or `df -kl` on Solaris and Linux operating systems. The `ovdbsetup` command asks you to enter the data directory and the index directory, defaulting to either `/u01/oradata/openview` or `/opt/oradata/openview`. If `/u01` or `/opt` are not existing file systems, the database will be created against `/` (slash) which typically is of insufficient size to successfully create the database.

To recover from a failure, see "If You Need to Rerun ovdbsetup" on page 56.

The script prompts you as follows:

```
Please enter the value for ORACLE_SID
Please enter the value for ORACLE_BASE
Please enter the value for ORACLE_HOME
Please enter the Oracle DBA user
Please enter the data directory
Please enter the index directory
Please enter Character set to use
```

Oracle uses the CHARACTER SET option of the CREATE DATABASE command to define the database language. When ovdbsetup creates the database, it defaults to the following character set for English language installations:

```
CHARACTER SET = "WE8ISO8859P1"
```

For NNM, other character sets may be used. However, other HP OpenView products, as well as third-party applications, may have more specific requirements about which character sets to use.

To create a Japanese database, use the following `CHARACTER SET` values:

`JA16SJIS` (for ShiftJIS)
`JA16EUC` (for EUC)

Another way to specify a Japanese character set is to start `ovdbsetup` with the Japanese character set option (`-j` or `-e`). Refer to the *ovdbsetup* online reference page (manpage on UNIX) for more information.

```
Please enter the password for Oracle user 'system'
```

```
Will you export topology data into
the Oracle openview database (y/n)? [y]
```

The script then prompts you to determine whether to export trend data to the relational database. (If trend data is exported, additional prompts appear to determine the disk space requirements for your trend data. Refer to “Calculating Disk Space Requirements for Trend Data Storage” on page 72.)

```
Will your installation export snmp trend
(data collector) data in the Oracle database (y/n)? [n]
```

```
Please enter the password for Oracle user 'ovdb'
```

NOTE You will need the user name and password to access the database. Keep this information.

ovdbsetup Tasks

After responding to the prompts, `ovdbsetup` completes the following tasks:

Phase 1 Runs `ovdbsetupo1.sh` to create the database (if it does not yet exist) by generating:

- Database directories

- Database `openview`

`ovdbsetupo1.sh` attempts to connect to an existing Oracle database with the instance supplied. If the connection is successful, Phase 2 is skipped and the installation prompts you to run Phase 3 (`ovdbsetupo3.sh`) manually.

Phase 2 Runs `ovdbsetupo2.sh` to continue the database configuration:

- Generates several tablespaces which are common to all NNM applications (roll-back tablespace, and temporary tablespace).
- Creates the Oracle data dictionary views.
- Creates additional rollback segments.
- Creates DBA synonyms.
- Creates the spool file, `crdbov2.lst`, for database configuration problems. This file is located in the directory that you specified as the data directory earlier in this process.

For specific details about the operations which `ovdbsetup` performs via the Oracle `svrmgrl` administration tool, consult the `$OV_BIN/ovdbsetupo2.sh` file included with this product.

Phase 3 Runs `ovdbsetupo3.sh` to complete the following tasks:

- Calculates the size of `OV_DATA` and `OV_INDEX` tablespaces and creates them, if disk space is available.
- Generates a default user (`ovdb`) for that database.

  Checks the connection from the user `ovdb` to the database `openview` using `SQL*DBA` (an ASCII line-based program for use in administering databases).
- Creates privileges and quotas.
- Creates and runs the `crdbnnm.sql` SQL script.
- Configures `Net8`. If the `listener.ora` and `tnsnames.ora` files do not exist, the files are created in the directory:

HP-UX:	`/etc`
Solaris:	`/var/opt/oracle`
Linux:	`$ORACLE_HOME/network/admin`

  See “Configuring Oracle Net8” on page 61 for more information.

- Starts the `TNS Listener`.
- `libclntsh.sl` (for HP-UX) or `libclntsh.so` (for Solaris and Linux) is recreated.
- Attempts to connect to the new Oracle database via the Oracle ODBC driver.
- Shuts down and restarts the NNM database in order to use the correct `init.ora` file.

  Changes the data warehouse configuration to use Oracle.
- Writes the most important information to the NNM `$OV_CONF/ovdbconf` file. (This enables `ovdbsetup` to handle multiple invocations and have the values available in the relational database installation scripts.)

  Sets the default environment variables in `$OV_CONF/analysis/ovdwenvars.conf`.
- Creates the data warehouse tables and indices in the database.
- Disables the embedded database.

NOTE For the data warehouse to work properly, you must run either **`ovdbsetup -o`** or **`ovdbsetupo3.sh`** if you manually created the database for the data warehouse.

NOTE NNM installs and configures the appropriate ODBC driver for the Oracle Relational Database. The ODBC data source is configured in the `$OV_CONF/analysis/system_odbc.ini` file.

If You Need to Rerun ovdbsetup

If problems occur during the execution of `ovdbsetup`, there are a number of actions you can take to reduce the recovery effort. `ovdbsetup` has 3 phases, each phase is contained in a subprogram. The phases are:

- `ovdbsetupo1.sh`

  Creates the desired Oracle database. The default name is `openview`.

- `ovdbsetupo2.sh`

  Creates the OpenView tablespaces and configures the instance for NNM and other HP OpenView applications.

- `ovdbsetupo3.sh`

  Configures NNM specific information for the `ovdb` user. Configures ODBC and sets up the data warehouse tables.

Rerunning a Portion of ovdbsetup

The `$OV_CONF/ovdbconf` file passes information between these subprograms. As long as the phases are run in sequence, a failure in one subprogram should only require that subprogram (and subsequent subprograms) to be rerun. For example, if `ovdbsetupo2.sh` fails because there was not enough disk space for the OpenView tablespaces:

1. Free enough disk space for the tablespaces. You may have to remove partially created files.
2. Rerun **`ovdbsetupo2.sh`**. You do not need to run **`ovdbsetup`** (including `ovdbsetupo1.sh`) from the beginning.
3. After `ovdbsetupo2.sh` finishes, execute **`ovdbsetupo3.sh`**.

Rerunning the ovdbsetupo3.sh Script to Recreate the NNM Tablespaces

CAUTION This procedure will delete the NNM tables containing the NNM data warehouse. All data in the NNM data warehouse will be lost.

Under some circumstances, you may desire to rerun the third phase of the `ovdbsetup` script in order to recreate the tablespaces for the NNM data warehouse. This phase is script `ovdbsetupo3.sh`.

1. Log in as user `root`.
2. Shutdown OpenView **`/opt/OV/bin/ovstop -c`**
3. Drop the NNM data warehouse tables:

   ```
   cd $OV_ANALYSIS_CONF/sqlScripts
   /opt/OV/bin/ovdwquery -file drop_event.oracle
   /opt/OV/bin/ovdwquery -file drop_trend.oracle
   ```

```
/opt/OV/bin/ovdwquery -file drop_topo.oracle
```

4. Run **`$ORACLE_HOME/bin/svrmgrl`** or $ORACLE_HOME/bin/sqlplus (depending on the Oracle release). Connect as user `system`.
5. Drop the NNM tablespaces:

```
SVRMGR> DROP TABLESPACE OV_DATA INCLUDING CONTENTS;

SVRMGR> DROP TABLESPACE OV_INDEX INCLUDING CONTENTS;

SVRMGR> exit
```

6. Remove the underlying NNM tablespace files on the file system:

`cd` ***`DATA_DIR`*** (the value of ***`DATA_DIR`*** is identified in `$OV_CONF/ovdbconf`)

```
rm OV_DATA.dbf
```

`cd` ***`INDEX_DIR`*** (the value of ***`INDEX_DIR`*** is identified in `$OV_CONF/ovdbconf`)

```
rm OV_INDEX.dbf
```

7. Rerun `ovdbsetupo3.sh` to recreate the NNM tablespaces for the data warehouse:

```
/opt/OV/bin/ovdbsetupo3.sh
```

TIP When this script is rerun, you will see the following error which can be safely ignored because the OVDB user has already been defined:

```
ORA-01920: user name 'OVDB' conflicts with another user
or role name
```

8. Restart OpenView:

ovstart -c

Rerunning All of ovdbsetup

CAUTION This section describes how to remove and recreate the `openview` database using the `ovdbsetup` tool. This tool would normally be rerun only in cases where the first attempt failed due to a condition such as insufficient disk space.

Following this rerun procedure will remove the entire `openview` database including any tablespaces belonging to other OpenView products. Therefore the `ovdbsetup` tool should not be rerun in an environment where multiple OpenView applications share the `openview` database.

NOTE If you wish to remove and recreate only the NNM tablespaces (including the NNM Data Warehouse tables), please refer to the previous section, "Rerunning the ovdbsetupo3.sh Script to Recreate the NNM Tablespaces."

If `ovdbsetup` must be completely rerun, then complete the following:

1. Log in as user `root`.
2. Stop the NNM processes by typing **`ovstop`**.
3. Shut down the Oracle `openview` instance.

   a. Execute the following command to determine if Oracle is running:

```
ps -ef | grep ora
```

   If Oracle is running, you will see 6 processes for:

```
ora_lgwr_openview
ora_smon_openview
ora_ckpt_openview
ora_dbwr_openview
ora_pmon_openview
tnslsnr
```

   b. Stop Oracle gracefully:

   # **`su - oracle`**

   # **`svrmgrl`** or **`sqlplus /NOLOG`** (depending on your Oracle release. This starts the server manager in line mode.)

   If you are running `svrmgr1`, at the server manager prompt, type:

   `connect internal`

   If you are running `sqlplus`, at the prompt, type:

```
connect system as sysdba
```

You will either get another prompt or a message saying you are connected. Both are acceptable. Now type:

```
shutdown normal
```

Several messages will display showing the stages of the database being shutdown.

```
Database closed.

Database dismounted.

Oracle instance shut down.
```

When you are returned to the prompt, type:

```
exit
```

c. Stop the Oracle Listener process:

Still logged in as the Oracle user (unix user), at the command line, type:

```
# lsnrctl stop
```

You should see messages similar to:

```
Connecting to (ADDRESS=(PROTOCOL=IPC)(KEY=openview)

The command completed successfully.
```

d. Exit from the oracle user (unix user) at the shell prompt.

4. Clean out the OpenView-created Oracle files.

a. Determine where the physical database files reside for the openview database:

```
# head -15 $ORACLE_HOME/dbs/initopenview.ora
```

Note the directory of the control files. This will identify where the database files that support NNM are located.

b. Change directories to the directory of the control files. From the example above, **`cd /opt/u01/oradata/openview`**

c. As root, remove files in the data and index directories. If `ovdbsetup` detects errors while creating the database, it attempts to determine which files should be removed before attempting to rerun. For instance, upon failing, `ovdbsetup` displays the following:

```
Then remove the following files from the
/u01/oradata/openview directory:
control01.ctl
control02.ctl
control03.ctl
redo01.log
redo02.log
redo03.log
system_1.dbf
temp_1.dbf
tools_1.dbf
RBS1_1.dbf
OV_DATA.dbf
OV_INDEX.dbf
```

d. Also execute the following commands to remove the rest of the NNM/Oracle dependencies:

```
rm /etc/opt/OV/share/conf/ovdbconf
rm -rf $ORACLE_BASE/admin/openview
rm $ORACLE_HOME/dbs/initopenview.ora
rm $ORACLE_HOME/dbs/lkOPENVIEW
```

Configuring Oracle Net8

The data warehouse requires Net8 to be configured. The `ovdbsetup` tool (`ovdbsetupo3.sh phase`) configures Net8 automatically if it does not detect existing `listener.ora` and `tnsnames.ora` files. The `ovdbsetup` tool provides a minimal environment suitable for the data warehouse.

NNM does not provide scripts to configure remote Oracle Net8 configurations; however, a knowledgeable Oracle database administrator can make the necessary modifications to set up remote configurations. Consult Oracle publications which explain remote configurations and how to customize your network for the best possible performance.

If you already have Net8 configured and want the data warehouse to use an Oracle database on a system remote from the NNM collection station, you must manually configure Net8. Sample configuration files are located in `$OV_CONTRIB/dataWarehouse`.

Consider the following tasks when you modify the Net8 environment:

1. Set the `KEY` parameter in the `listener.ora` file equal to the `ORACLE_SID` for the database instance you are using.

2. Set the HOST and PORT parameters in listener.ora to define the TCP service where your Oracle server is located.
3. In the tnsnames.ora file, define a connection for ov_net.
4. If you want to specify a different connection than that identified in ov_net, you must change the ServerName parameter in the ODBC configuration file:

   $OV_ANALYSIS_CONF/system_odbc.ini.

   NOTE: If you are using other HP OpenView products which also use Oracle, you must make the appropriate changes for the other HP OpenView products.
5. The Net8 listener process must be started by user oracle before starting the NNM processes with ovstart. Enter:

   su - oracle

   lsnrctl start
6. To verify the Net8 connection:

   a. Set the $ORACLE_HOME environment variable.

   b. Start **sqlplus**. Enter:

      $ORACLE_HOME/bin/sqlplus ovdb@ov_net

   c. Enter the password for the database user ovdb.

      A set of messages display indicating you are connected to the data warehouse Oracle database instance.

Starting and Stopping Oracle

If you are running Oracle with data warehouse, you must manually perform the necessary configuration steps to have the Oracle database server and the Net8 listener components started *prior* to the NNM startup tool, ovstart. If these Oracle components are not available, ovdbcheck will fail to start.

Oracle provides the dbstart and dbstop tools to start and stop an Oracle instance. Refer to the Oracle Installation Guide specific to your Oracle version and operating system for information on updating your oratab file for the openview instance and configuring these tools in the /sbin/init.d directory to be automatically executed at system startup and shutdown.

Number the startup script as low as possible so that it starts before NNM is started. Number the kill script as high as possible so that Oracle is shut down after NNM.

You have completed the installation and configuration of Oracle on a UNIX system.

Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems)

You can install the Oracle database as a stand-alone workstation server or on a remote server. Refer to your Oracle documentation for information about the Oracle products. The products shown below are required to run together with NNM.

NOTE See the Release Notes, Supported Configurations topic for information on supported versions of these databases.

- Oracle server
- SQL*Plus
- SQL*Net
- TCP/IP protocol adapter for SQL*Net
- The following components are not required by NNM, but can be used for your customized reporting:
  - Oracle Data Query
  - SQL*ReportWriter
  - SQL*Report

Configuring Oracle

If you do not have Oracle installed, follow the Oracle installation guide to install the products required by NNM.

You may choose to create an `openview` database after Oracle is installed or allow Oracle to create a default database to which you add the necessary objects. At a minimum, you should create the following objects:

- `OV_DATA` tablespace
- `OV_INDEX` tablespace

- `TEMP` tablespace
- OVDB user with the following characteristics:
  - Default tablespace is `OV_DATA`
  - Default temporary tablespace is `TEMP`
  - Unlimited quota on `OV_DATA`, `OV_INDEX`, and `TEMP`
  - Connect and resource privileges

Oracle Network Configuration

Use Net8 Easy Config to create a new service for the NNM data warehouse. HP recommends using `openview` for the service name (also called a database alias). The database SID (System Identifier) must match the database created above.

Setting Up ODBC Data Source

NNM preinstallation and installation procedures install and configure the appropriate ODBC driver and ODBC Driver Manager for the embedded database. If you want to use the Oracle Relational Database product, you must reconfigure the data warehouse.

Follow these steps to create an ODBC data source for your Oracle `openview` database:

1. Launch the ODBC Administrator from the Control Panel.
2. Choose the `System DSN` tab.
3. Click `[Add]`.
4. Select the Oracle driver.
5. Click `[Finish]`.
6. Complete the setup as follows:
   - Data Source Name: user-defined, for example OVoracle.
   - Description: user-defined, for example NNM to Oracle ODBC.
   - Service Name (known as "connect string" in SQL*Net): Enter the service name (also known as a database alias) from your Oracle Network Configuration, for example `openview`.
   - User Id: Oracle user created when configuring Oracle, for example `ovdb`.

7. Click `[OK]`.

Configuring for the Data Warehouse

To configure the data warehouse to use the Oracle relational database you have installed, enter the following command:

```
drive:\install_dir\bin\ovdwconfig.ovpl -rdb ODBC_datasource
-u userid -password password -load -type oracle
```

where

- *drive* is the drive letter where NNM is installed.
- *install_dir* is the directory where NNM is installed. If this directory contains spaces then you must specify the directory path within quotes.
- *ODBC datasource* is the data source name specified in the ODBC setup, for example OVoracle.
- *userid* is the Oracle user created under Configuring Oracle, for example, `ovdb`.
- *password* is the password of the Oracle user specified in the userid.
- The `-load` option is used for a newly-configured data warehouse. It creates the schema for the `openview` database.

You have completed the installation and configuration of Oracle on a Windows operating system.

Removing the NNM Database Objects

Should it become necessary to remove the NNM data warehouse Oracle objects, follow this procedure.

NOTE This removes all objects within these tablespaces as well as the tablespaces themselves. HP recommends a database backup prior to making significant changes.

1. Remove the Oracle `ovdb` user with the `cascade` option. This removes all tables, indexes, permissions created as part of the `ovdwconfig.ovpl` script (This was run when you initially configured the data warehouse to use Oracle as its external database.)

2. Alter the tablespaces you created to support the NNM data warehouse so that they are off-line.
3. Once the tablespaces are off-line, drop the tablespaces.
4. Remove the operating system files associated with the tablespaces.

Installing and Configuring Microsoft SQL Server

Refer to the Microsoft SQL Server documentation for specific installation procedures. This section describes the configuration of the Microsoft SQL Server Relational Database product.

Configuring Microsoft SQL Server

After the installation is complete, configuration of the Microsoft SQL Server consists of creating the following objects (refer to the SQL Server documentation for specific instructions):

- Database devices for the `openview` database and log:
  - HP recommends that you create separate devices for the database and the log.
  - You can calculate your sizing information from "Calculating Disk Space Requirements for Trend Data Storage" on page 72. The recommended value for the `OPEN OBJECTS` parameter is 10,000.
  - When determining the file location of the device, do *not* choose `\`*`install_dir`*`\databases`. A directory choice on a different physical disk device minimizes I/O contention during exports.
- A database for the OpenView objects. HP recommends using the name `openview`.
- An `openview` database user. The recommended name is `ovdb`.
- Grant the `openview` user permissions on the `openview` database to:
  - create table
  - create sp
  - create default
  - dump DB and dump trans (depending on your backup strategy)

Setting Up the ODBC Data Source

NNM preinstallation and installation procedures install and configure the appropriate ODBC driver and ODBC Driver Manager for the embedded database. If you want to use the Microsoft SQL Server product, you must reconfigure data warehouse.

Follow these steps to create an ODBC data source for your Microsoft SQL Server `openview` database:

- Launch the ODBC Administrator from the Control Panel.
- Choose the `System DSN` tab.
- Click `[Add]`.
- Select the SQL Server driver.
- Click `[Finish]`.
- In the resulting screen, click `[Options]`.
- Complete the set up as follows:
  - — Data Source Name: user-defined (for example OVSQL).
  - — Description: user-defined (for example, NNM to SQL Server ODBC).
  - — Database Name: as defined above (for example `openview`).
  - — Complete other fields as needed.
- Click `[OK]`.

Configuring for the Data Warehouse

To configure the data warehouse to use the Microsoft SQL Server relational database you have installed, enter the following command:

```
drive:\install_dir\bin\ovdwconfig.ovpl -rdb ODBC_datasource
-u userid -password password -load -type msSqlSrvr
```

where

- *`drive`* is the drive letter where NNM is installed.
- *`install_dir`* is the directory where NNM is installed.
- *`ODBC datasource`* is the data source name specified in the ODBC setup, for example OVSQL.

- *`userid`* is the database user created above, for example `ovdb`.
- *`password`* is the password of the database user specified in the userid.
- The `-load` option creates the schema for `openview` databases. It is required for new installations.

You have completed the installation and configuration of Microsoft SQL Server on a Windows operating system.

Validating NNM's Connection to a Relational Database

Installing a relational database into the NNM environment does not make the database available to connect the data warehouse. The database must be configured for use by NNM applications.

You can use the optional `ovdbdebug` utility to determine whether the database is up and able to connect to the data warehouse.

- If you are running `ovdbsetup`, it automatically performs this validation at the end of its configuration steps by running `ovdbdebug` for you.
- If you are not using `ovdbsetup`:
  - For UNIX, run **`ovdbdebug`** or **`ovdbcheck`**.
  - For Windows operating system, run **`ovdbcheck`**.

Calculating Disk Space Requirements for Trend Data Storage

Since a comprehensive set of algorithms for the precalculation of required disk space is not practical, the following discussion provides some guidelines for determining disk space requirements for the data warehouse databases.

Due to the use of "varchar" (variable character) data types in the database, the number of bytes of data in each row of the database will be highly variable. The following section describes how the SNMP trend data tables are constructed, so you can approximate how much disk space is required for the amount of SNMP data you intend to collect.

Trend data is collected and stored cumulatively, a process which produces a large amount of data. The data can very quickly consume large amounts of disk space and CPU resources.

For an Oracle Relational Database

Since Oracle pre-allocates disk space, `ovdbsetup` on UNIX determines your trend disk space needs. `ovdbsetup` sets up tablespaces for use in data storage. NNM data is stored in the `OV_DATA` tablespace. Indexes are stored in the `OV_INDEX` tablespace, if it exists.

If you do not use `ovdbsetup` to create the database, the following information may be helpful to calculate your trend data storage needs:

- A reduced trend data record uses approximately 110 bytes, which includes space for the record's index.
- A raw trend data record uses about 80 bytes, which includes space for its index.
- You can estimate the amount of trend disk space needed for `OV_INDEX`, `OV_DATA`, and any `TEMP` tablespaces by manually running the `ovdbtrend_dsko.sh` utility.

For NNM's Embedded Database

When data is stored in NNM's embedded database, it requires about the same amount of disk space as it would if stored in Oracle.

However, storage consumption increases over time, not all at once. Due to the way the embedded database gets backed up, it requires three times the disk consumption for the same number of collections/events.

By default, this all ends up in `/var` along with the other files NNM accumulates throughout time.

Administrators are advised to keep a watch on the disk consumption in `/var`, as this is where NNM's data ends up. It is not uncommon to exhaust space in `/var`.

Considerations for Calculating Disk Space Requirements

To calculate the amount of disk space needed for your database to store all your trend data records, follow the procedure below:

1. Estimate the data collections that are (or will be) configured, with either method below.

   - If you are already collecting the values to be exported to your database, complete the following tasks while `snmpCollect` is running:

     a. Execute the **`snmpCollect -S`** command.

     b. Check the `$OV_LOG/snmpCol.trace` file to determine the current number of collections. Look at the `Per Collection Information` section of the configuration display.

   - Use the following formula for your calculations:

     `(number of mib variables)`
     X *`(average number of instances per variable)`*
     X (*`average number of nodes collected upon per variable`*)
     = *`number of collections`*

2. Estimate the number of records to be collected each day and the length of time these records will remain in the database.

   a. Use the formula below to estimate the number of records collected daily:

      1440 / *`collection interval in minutes`*
      = *`number of collections per day`*

where 1440= the number of minutes per day and *collection interval in minutes* = either the reduction interval (for data stored in reduced format) or the actual collection interval (for data stored in raw form).

b. Determine the length of time (in days) that records will remain in a database.

3. Use the results from Steps 1, 2a, and 2b to calculate the maximum number of trend data records to be stored in the database.

   (*number of collections*, from Step 1)
   X (*number of collections per day*, from Step 2a)
   X (*number of days stored*, from Step 2b)
   = number of records

4. Determine the total disk space needed in the database for trend data:

   (*number of records*, from Step 3)
   X (*number of bytes used per record*)
   = *total disk space*

   where *number of bytes used per record* uses the following values:

   — A reduced trend data record stored in the default btree storage structure occupies approximately 135 bytes.

   — A reduced trend data record stored in a hash storage structure occupies approximately 110 bytes.

   — A raw trend data record stored in the default btree storage structure occupies approximately 110 bytes.

   — A raw trend data record stored in a hash storage structure occupies approximately 70 bytes.

NOTE The amount of disk space calculated from this procedure addresses the space needs for only the trend data and indexes.

Some operations, such as data table modifications and queries, can require significant amounts of disk space for sorting purposes. For example, if you modify a database table (such as when the btree or hash structure is initially imposed on trend data tables), you may need three times the actual data table size during its modification. In the case of queries, it is recommended that the free disk space is at least half the maximum size of the trend data table being accessed.

Example Calculation of Trend Data Disk Space

The example below illustrates the calculation of trend data disk space. Formulae are shown in italics, and calculations are shown in bold:

1. Determine the number of configured data collections:

   (*number of mib variables*)
   X (*average number of instances per variable*)
   X (*average number of nodes collected upon per variable*)
   = number of collections

   (12 MIB variables)
   X (average of 10 instances per variable)
   X (100 nodes)
   = 12000 collections

2. Determine the number of records to be collected each day and the length of time these records will remain in the database.

   a. Estimate the number of records collected daily:

      1440
      / *collection interval in minutes*
      = *number of collections per day*

      1440
      / 60
      = 6

      where 60 = the number of minutes in the default reduction interval of one hours.

   b. Determine the length of time (in days) that records remain in a database. For this example, trend data older than **30 days** will be deleted.

3. Use the results from Step 1, 2a, and 2b to calculate the maximum number of trend data records to be stored in the database.

(*number of collections*, from Step 1)
X (*number of collections per day*, from Step 2a)
X (*number of days stored*, from Step 2b)
= number of records

12000
X 24
X 30
= 8,640,000 records

4. Finally, determine the total disk space needed in the database for trend data:

   (*number of records,* from Step 3)
   X (*number of bytes used per record*)
   total disk space

   8,640,000
   X 135
   = 1,166,400,000 bytes of total disk space (approximately 1.2 GB)

   where 135 = the size of a reduced trend data record stored in the default btree storage structure.

Transaction Log Requirements

The transaction log requirements vary, depending on the use of export utilities and the methods used to access the trend data.

HP recommends a minimum transaction log size of 32MB for storing trend data.

Disabling and Enabling the Embedded Database and Web Reporting Interface

During the configuration of the NNM-supported database products for the data warehouse, the `ovdwconfig.ovpl` command automatically reconfigures `ovdbcheck` to NOT start the embedded database server.

If the data warehouse functionality is not required or wanted, you can permanently disable the embedded database by using the `ovdelobj` command.

CAUTION If you purchased the NNM Advanced Edition and enabled the Extended Topology functionality, *do not execute the* ***`ovdelobj ovdbcheck.lrf`*** *command from the following procedure to disable the* `ovdbcheck` *process.* Using this command to disable the `ovdbcheck` process is not supported for NNM Advanced Edition with Extended Topology enabled.

If you purchased the NNM Advanced Edition and enabled the Extended Topology functionality, you can use the **`ovdelobj ovrequestd.lrf`** command in following procedure to disable the reporting functionality.

To disable the embedded database:

1. Go to the *`install_dir`*`\lrf` (for Windows operating systems) or `$OV_LRF` (for UNIX) directory.
2. Execute the commands:

   ```
   ovdelobj ovrequestd.lrf
   ```

   ```
   ovdelobj ovdbcheck.lrf
   ```

   After these commands are executed, `ovstart` will not start `ovdbcheck` or `ovrequestd`. `ovrequestd` schedules the reporting tasks.

To enable the database at a future time:

1. Go to the *`install_dir`*`\lrf` (for Windows operating systems) or `$OV_LRF` (for UNIX) directory.
2. Execute the commands:

```
ovaddobj ovrequestd.lrf
```

```
ovaddobj ovdbcheck.lrf
```

For more information, see *ovaddobj*(1) manpage or reference page.

4 Migrating NNM Data Between Databases

This chapter describes the methods for migrating existing NNM data to a different database. The migration procedures are different for each type of data (topology, trend, or event).

To migrate NNM data from versions prior to 6.0 to the current version of NNM, see the *NNM Migration Guide*.

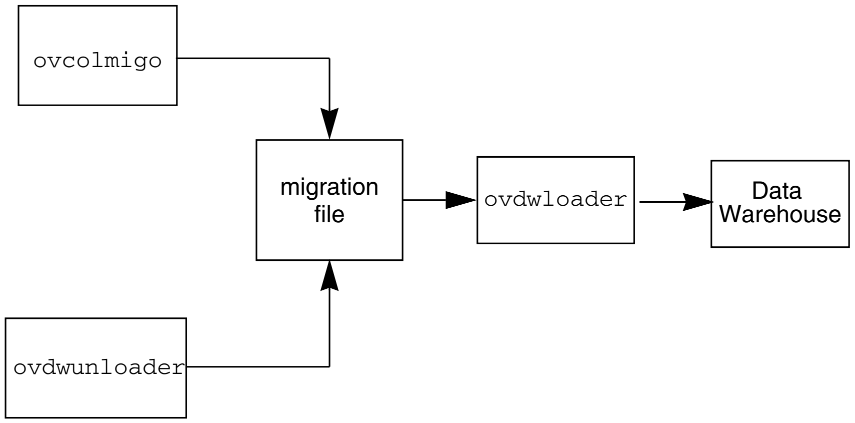
Migration Considerations

Migrating your NNM data between different relational database products, as well as between different releases of a particular database product, is supported by the following data warehouse commands:

- `ovdwconfig.ovpl`—changes the data warehouse configuration to support any of the NNM-supported relational databases
- `ovcolmigo`—converts the trend data from existing Oracle databases (before NNM 6.0) and puts it into a migration file
- `ovdwloader`—loads the migration data back into the data warehouse
- `ovdwunloader`—unloads data from the data warehouse to migrate to another database

Refer to the *ovdwconfig.ovpl*, *ovcolmigo*, *ovdwloader*, *ovdwunloader* manpages or reference pages for more information about these conversion utilities. Figure 4-1 illustrates the migration of data.

Figure 4-1 **Migrating Data to Data Warehouse**



NOTE CA-Ingres is no longer a supported database for use with NNM. All data stored in CA-Ingres must be migrated to another relational database. See the *NNM Migration Guide*.

If you have existing scheduled export and trim tasks and `cron` jobs (on UNIX systems) using the commands `ovcoltosql` and `ovcoldelsql`, your first choice is to replace them with web Reporting interface reports. If those do not meet your needs, your scripts should be updated to use the current NNM tool, `ovdwtrend`.

Migrating Oracle Data to NNM 6.x

Topology Data Migration

You do not need to migrate the topology data from an existing Oracle or CA-Ingres database to another relational database. By default, the topology data is exported to the data warehouse every 12 hours.

Trend Data Migration

Due to changes that were made in the data warehouse schema for NNM 6.0, you must migrate trend data collected by earlier versions. At any time after NNM installation, to migrate existing trend data in an existing Oracle database to the data warehouse, follow the steps below:

1. Set the Oracle environment variables such as `$ORACLE_HOME`, `$ORACLE_BASE`, and `$ORACLE_SID`. See the Oracle documentation for more information about these environment variables.
2. Enter:

   `su -ovdb`

3. Convert the data (stored in databases from previous NNM versions) from the relational database to a migration file. Execute the command:

`ovcolmigo -v -f` ***`filename`***

The current data is moved from an Oracle database into a migration file. The *`filename`* is the output file that contains the migration file. The output file requires approximately the same amount of free space on the file system as the original Oracle database. The data in the following tables is exported and converted, as necessary:

- `snmp_trend_dataset`
- `snmp_trend_descript`
- `snmp_reduced_trend`
- `snmp_raw_trend`

4. Choose the Oracle instance to run with the data warehouse:

   a. Backup the Oracle database (see Chapter 9, Backup and Recovery,).

   b. Remove the old `ovdb` system login. It is no longer used for operating system authentication

   c. Login as `root`, then enter:

      `ovdbsetup -o`

      - If `ovdbsetup` finds an existing Oracle instance, it sends the prompt, `Do you want to use this Oracle instance?`

        Answer yes to use this Oracle instance, then run **`ovdbsetupo3.sh`**.

      - If `ovdbsetup` does not find an existing Oracle instance, it asks whether you want to create a new instance.

        Answer yes to create a new Oracle instance. `ovdbsetup` automatically executes all three phases of the configuration.

5. Load the migration file into the data warehouse. Execute the command:

```
ovdwloader -v -file migrationfile
```

During this conversion process, any necessary schema changes are made.

Event Data Migration

It is not possible to migrate event data obtained prior to NNM version 6.0 to the data warehouse.

Migrating Data Between Database Products

Current NNM SNMP trend and event data can be transferred between database products. For example, to move data from the embedded database to another database:

1. Make sure `ovdbcheck` is the only process that is running.
2. Unload the data by executing one of the following commands:

`ovdwunloader -file` ***`outputfile`*** **`-trend -v`**

The current trend data is moved from the database into a migration file.

`ovdwunloader -file` ***`outputfile`*** **`-event -v`**

The current event data is moved from the database into a migration file.

3. Install the new relational database and configure the ODBC datasource. See Chapter 3, "Installation and Configuration."
4. Run the following command to configure the tables in the database.

`ovdwconfig.ovpl -u` ***`userid`*** **`-password`** ***`password`*** **`-type oracle`**
`-rdb` ***`ODBC_datasource`*** **`-reload`**

The `-reload` option removes the existing data warehouse tables and the data within the tables. You should use this option to obtain the best results.

5. Reload the data from the migration file ***`outputfile`*** to another database product using the command:

`ovdwloader -file` ***`inputfile`***

The data in ***`inputfile`*** is loaded into the data warehouse. Multiple tables can be loaded during one execution of `ovdwloader` by concatenating each table together into ***`inputfile`***.

NOTE Because the topology data is a replication of the data in the operational database, no specific migration functionality is supplied.

5 Web Reporting Interface

This chapter describes how to use NNM's web Reporting interface utilities.

In addition to being able to write your own SQL queries of the data warehouse, NNM provides automatic report generation through the web Reporting interface to facilitate managing your network. When you configure automatic report generation, NNM queries the data warehouse for you, and can even e-mail the results back to you regularly.

Configuring Reports

Network Node Manager provides several predefined report templates which you can customize and set up for automatic report generation through the NNM Report Configuration GUI.

When you activate a report, several processes are affected:

- For reports based on SNMP data, data collection is started for the report.
- The data warehouse receives data. The RDBMS you use is transparent to the report generation process.
- At periodic intervals, summarization of data occurs in the data warehouse.
- The web Reporting interface generates a report, e-mails the report to a list of recipients, and stores the report for viewing.

You can list e-mail addresses to which the reports are sent. Users can also view the generated reports through the NNM pull-down menu `Tools:Report Presenter`, or through the web Launcher under the `Information and Reports` tab.

Using the Report Configuration Interface

Use the Report Configuration interface to create, modify, view report schedules and delete reports. You can access the Report Configuration interface in different ways:

- From the NNM menu, select `Options:Report Configuration`. The `Create NNM Reports: NNM Analysis and Reporting` window appears.
- From the Launcher:

    1. Select the `Tasks` tab.
    2. Expand the `Information and Reporting` category.
    3. Double-click on an item to create, modify, or view the report schedule.

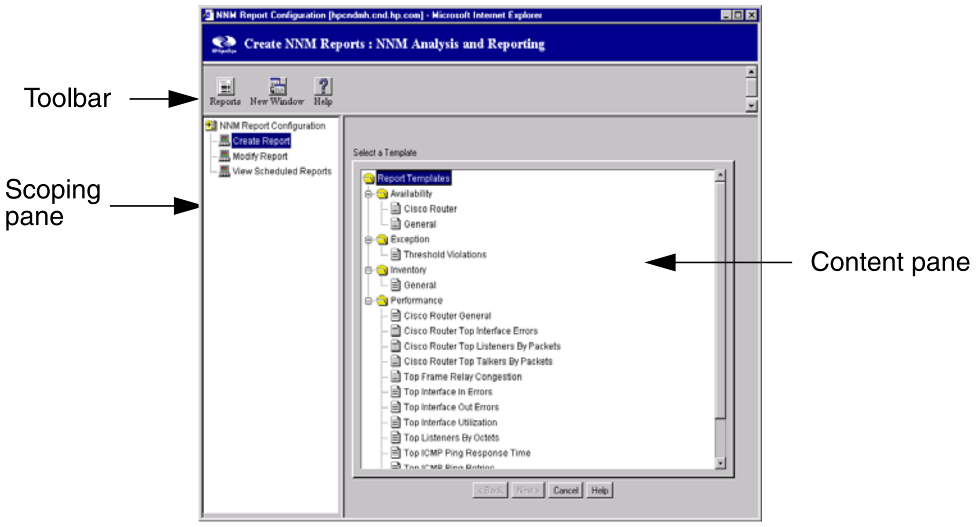
    OR

1. Select the `Tools` tab.
2. Expand the `NNM` category.
3. Double-click on `Report Configuration`.
4. Double-click on a category from the scoping pane to create, modify, or view the report schedule.

- You can also access Report Configuration from the `Options` menu of the Network Presenter.
- Bookmarking the Report Configuration interface is another easy way to access Report Configuration. Use the following URL for your bookmark:
  `http://`*`hostname`*`[:`*`port`*`]/OvCgi/nnmRptConfig.exe`

  Where the port is necessary only when connecting to a UNIX server, and the customary value is 3443.

Figure 5-1 Report Configuration User Interface



Toolbar

The toolbar contains three elements, `[Reports]`, `[New Window]`, and `[Help]`. The `[Reports]` button displays the Report Presenter where the administrator or user can view the current reports. The `[New Window]` button clones the current window. The `[Help]` button displays the online help.

Scoping Pane

Choose the type of configuration to do in the scoping pane. You can create a new report, modify an existing report, or view the list of scheduled reports.

Content Pane

When you are creating a report, the content pane lists all the report templates available. When you are modifying reports or viewing scheduled reports, only scheduled reports are presented. Choose a report by double-clicking on the report name. This displays the configuration GUI for that report.

Using the Predefined Templates

A set of predefined templates are shipped with NNM. These templates define the type of data to be included in a report and the final presentation of the report. Templates are customized with user-specific data through the Report Configuration user interface.

Detailed information for all the templates is available through online help.

Availability Report Templates

Availability reports list the percentage availability of the node interfaces on each network, with hyperlinks to the detailed reports for the interfaces.

Exception Report Template

The exception report shows threshold events which have been exceeded. You can configure specific threshold events at each node, which may be different between nodes. When any threshold is exceeded, it is added to this report. The report also contains hyperlinks to detailed reports for the specific thresholds.

Inventory Report Templates

The inventory report tells you what nodes are in the topology database on a daily basis.

Performance Report Templates

The performance reports appear as summary tables for the interfaces. The values are the mean values over the period of the report. You may also click on the interface to see a graph of its historical data.

Performance reports that are not Cisco-specific differ from the Cisco reports in that they choose their data collection and calculation based on the nature of the interface, including duplex, packet orientation, and capacity.

For example, the “Cisco Router Top Interface Errors” reports the 10 Cisco interfaces with the highest error rate. Values are from the Cisco MIB and include interfaces which are down. The “Top Interface In Errors” reports on the 10 interfaces with the highest percentage of inbound packet errors. Data is collected from MIB II expressions, according to the type of interface, and excludes any interfaces which are down.

Steps to Configure a Report

There are eight steps to configuring a report:

1. Select the nodes. (For reports based on SNMP collected data only)

   Specify the target nodes.

   You can specify hostnames, IP addresses, IP address wild cards, interfaces, and node filter files. For each node target, all interfaces are assumed, unless interfaces are specified.

2. Select the data. (For reports based on SNMP collected data only)

   For reports based on SNMP collected data, specify the data collection intervals. Default values are provided. The data collection is the name of the MIB expression, such as `%InPackets`, on which the data is being collected. The data collection interval can be 5, 10 or 15 minute intervals.

3. Define the report generation schedule.

   Scheduling the reports defines both the frequency of generation and the time span of the report. The available values are daily and month-to-date.

4. Name the Report.

   Give the report a unique name. For example, Daily Inventory. Defaults are provided, but may be confusing if multiple operators schedule similar reports.

5. Specify e-mail addresses.

   You can list e-mail addresses. The use of a mail alias is recommended. This allows you to change the distribution list without having to modify report configuration. When the report is generated, the Report Presenter sends the HTML report to the addresses. To specify multiple addresses, use either a space or a comma as the delimiter.

6. Exit the configuration wizard. NNM saves the new report automatically.

   Saving the report activates the new report. Once the report is activated, the reporting processes start collecting data and storing the data in the data warehouse.

7. Configure thresholds on the MIB expressions of interest to you.

   NNM Exception reports include any threshold violation events that you have configured to trigger the standard `OV_THRESHOLD_VIOLATION` event identifier. Threshold collections that you have configured to use custom event configurations are not included in NNM web reports.

   NNM configures SNMP data collection on its own MIB expressions for web report generation for Performance reports. To set the threshold and rearm values for these expressions:

   a. Select the `NNM` menu item `Options:Data Collection & Thresholds:SNMP`.

   b. Scroll down to the expressions with an origin of `Reporting`.

   c. Select the expression for the current report and edit it to set threshold and rearm values.

   d. Verify that the status of the expression is `Collecting`.

Viewing Reports

Once the reports are generated, the user can view them in one of four ways:

- The report is mailed to selected e-mail addresses.
- Using the NNM pull down menu `Tools:Report Presenter`.
- Through the Launcher.

  1. Select the `Information and Reports` tab.
  2. Expand a category to see if the NNM Reports item appears.
  3. Double-click on `NNM Reports` to see if reports are available for viewing.

  OR

  1. Select the `Tools` tab.
  2. Expand the `NNM` category.
  3. Double-click on `Report Presenter`.

- Bookmarking the Report Presenter is another easy way to access this interface. Use the following URL for your bookmark: `http://`*`hostname`*`[:`*`port`*`]/OvCgi/nnmRptPresenter.exe`.

The `View NNM Reports:NNM Analysis and Reporting` window appears.

Suspending a Report

To stop a report for a temporary period, use the `[Suspend]` button. Suspending a report keeps the report in the scheduled list. The data collection for the suspended report continues and NNM continues to export data to the data warehouse. To reactivate the report, click on `[Resume]`.

You can modify a report when it has been suspended, but not when it has been stopped.

Use the following procedure to suspend a report from the configuration GUI:

1. From the scoping pane of the NNM `Reports:NNM Analysis and Reporting` window, double-click on `View Scheduled Reports`.
2. Select a report that you want to suspend.
3. Click on the `[Suspend]` button.
4. Click the `[Cancel]` button to cancel the operation and return to the list of scheduled reports.

Modifying a Report

Use the following procedure from the Report Configuration tool to modify a report:

1. From the scoping pane of the NNM `Reports:NNM Analysis and Reporting` window, double-click on `Modify Scheduled Reports`. The content area displays the `Select a Report` tab. This tab contains a set of existing reports that you can select from.
2. Double-click on a report or select a report and click `[Apply]`. The content area displays a tabbed property sheet that allows you to modify the selected report.

Stopping a Report

When no one any longer needs a particular report, you can stop the report.

Stopping a report does not automatically delete existing (stored) reports. A pop-up dialog box is activated when you select `[Stop]`. Optionally, you can delete the old reports at this time. If you choose to stop, but not remove, the report, you can use the operating system command line to manually delete the report files at a later time.

Stopping a report changes and may halt data collection. If only one report is being generated, and no other report is using similar data, the reporting processes stop exporting the data. For example, if a monthly report of the topology is being generated, and no other report uses topology data, then if you deactivate this report, the topology data is no longer collected and stored in the data warehouse.

When you stop a performance report, the reporting processes stop collecting the data for that report and stop generating the report. Other trend data associated with other configured reports is still exported into the data warehouse at periodic intervals.

Use the following procedure to stop a report from the configuration GUI:

1. From the scoping pane of the NNM `Reports:NNM Analysis and Reporting` window, double-click on `View Scheduled Reports`.
2. Select a report that you want to stop.
3. Click on the `[Stop]` button. A pop-up window appears, asking you if you wish to remove the report, stop the report but not remove it, or cancel the stop operation.
4. Click the `[Remove]` button to remove all the previously generated reports from the web Reporting interface. You will need to recreate the report using the Create Report wizard to generate this report again.

   Click the `[Do Not Remove]` button to not remove the report from the Report Presenter. You can reactivate the report by double-clicking on `Modify Report` from the scoping pane, selecting the report, and clicking on the `[Apply]` button.
5. Click the `[Cancel]` button to cancel the operation.

Deleting a Report

The reporting processes that create and store reports do not delete any reports. Once you determine the old reports are no longer useful, you need to delete them explicitly.

When you delete a report from the configuration GUI, you must use the Create Report wizard to configure the report again. Use the command line to manually delete older reports without having to recreate them.

NOTE Once a report has been deleted from your disk, there are no tools to recreate it from the data left in the data warehouse.

Use the following procedure to delete a report:

Using the GUI

1. From the scoping pane of the NNM `Reports:NNM Analysis and Reporting` window, double-click on `View Scheduled Reports`.
2. Select a report that you want to delete.
3. Click on the `[Stop]` button. A pop-up window appears, asking you if you wish to delete the report, stop the report but do not delete it, or cancel the stop operation.
4. Click the `[Remove]` button to stop the report from being generated and remove the report from the Report Presenter. This removes all previously generated reports from the web Reporting interface.
5. If you want new instances of the report to be generated, reschedule the report.

Using the Command Line

Reports are stored under the directory

Windows:
`*install_dir*\www\htdocs\C\nnm\reportPresenter\Reports`

UNIX: `$OV_SHARE_HTDOCS/C/nnm/reportPresenter/Reports`

Under this directory, the reports are in subdirectories that correspond to the categories of reports. For example, the General report from the Performance Top Interface Utilization template would be in the directory

Windows: `install_dir\www\htdocs\C\nnm\reportPresenter\Perf\Top+Interface+Errors\`

UNIX: `$OV_SHARE_HTDOCS/C/nnm/reportPresenter/Perf/Top+Interface+Errors/`

Under this directory, would be another directory for each generated report. For example `daily/daily.19990306/` on UNIX or `daily\daily.19990306\` on Windows operating systems.

To delete a single report, execute the following command:

Windows:

```
del daily.19990306\*.*
```

```
rmdir daily.1990306
```

UNIX: **`rm -rf daily.19990306`**

Using the command line to delete a report does not stop the report from being generated. Therefore you do not need to use the Create Report wizard to recreate the report.

NOTE For localized reports, NNM creates the report in the locale directory (`$LANG`) based on the language in use in the browser (Preferred Browser Locale) when the report was created. If this language is something other than English (the `../C/` directory), a “`C`” report is also generated. To remove such a report, you must remove both the `$LANG` and the “`C`” versions.

6 Export Utilities

The NNM data is stored in NNM databases. The data must be exported to the data warehouse prior to analysis and reporting. This chapter describes the utilities provided by the data warehouse for data export:

- `ovdwtopo`
- `ovdwtrend`
- `ovdwevent`

Explicitly Exporting Data to the Data Warehouse

NNM contains data that is useful for making decisions about network management. To access this data, it must first be exported to the data warehouse. In the data warehouse, tools are provided to aggregate and trim the data prior to generating reports.

The data warehouse provides several utilities for reporting data summaries. These utilities can be used from:

- The command line.
- A scheduler program such as `cron` on UNIX systems and MKS Scheduler or Microsoft SQL Enterprise Manager on Windows® operating systems (see "Using a Scheduler" on page 117 for detailed information).
- The `Tools` menu from the NNM pull-down menus.

  To access the utilities, from the menu bar, select `Tools:Data Warehouse->Export Topology` or `Export Event` or `Export Trend Data`. These export utilities are run without any additional command line options. For example, the menu bar selection `Tools:Data Warehouse->Export Topology` runs the command `ovdwtopo -export`.
- The web Reporting interface. These processes automatically schedule data collection, data export, and report formatting.

NOTE If the web Reporting interface reports do not meet your needs, the recommended use for the data warehouse export utilities is via a scheduler program. Although the `Tools` menu is a simple method for running the export utilities, the commands are run by default with the default command settings.

To change the default settings for the export commands from the `Tools` menu, modify the data warehouse application registration file. See *Creating and Using Registration Files* for more information about registration files.

A job scheduler can be used to periodically export and trim the appropriate data stores. Refer to “Using a Scheduler” on page 117 for more information.

Managing Topology, Trend, and Event Data

Three databases store NNM data. The data warehouse utilities are used to export data from the NNM databases, store and manage the data. The following utilities are used to access the data:

- The topology database stores the topology information.

  The utility, `ovdwtopo`, enables you to export the entire topology database to the data warehouse.

- The SNMP trend data is stored in a binary database.

  The utility, `ovdwtrend`, enables you to export, summarize, and trim trend data and delete data from the binary database.

NOTE

Periodically, you may need to delete NNM `snmpCollect` data. You can complete this task manually, using the NNM-provided tool, `ovdwtrend -delpriorto`, or writing your own tools.

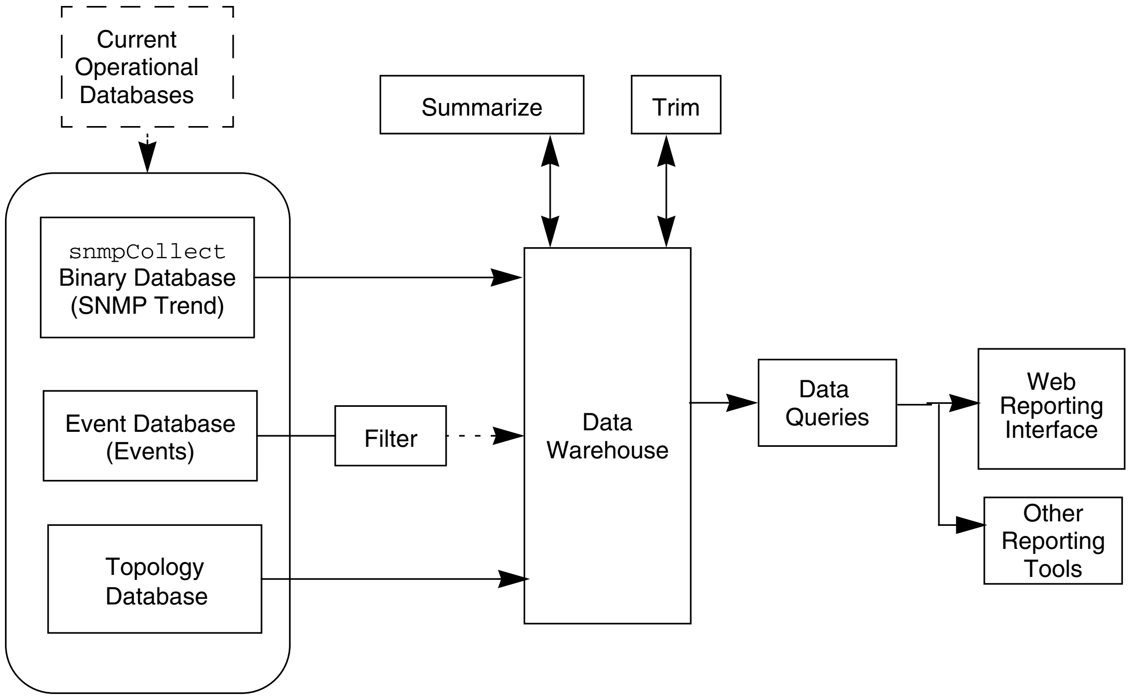
Data collected for web Reporting interface reports is automatically summarized into `DAILY` and `WEEKLY` tables, and data that is more than 7 days old is removed from the `REDUCED` tables.

- Events are stored in the event database.

  The utility, `ovdwevent`, enables you to export, trim, and summarize event data.

NNM collects the data and stores it in the operational databases. The data warehouse utilities export the data to the data warehouse database. Once the data is in the data warehouse, you can query the data to generate reports. This data flow is illustrated in Figure 6-1.

Figure 6-1 Flow of Data from NNM to Data Warehouse to Reports



The data export and management tools are described in the sections that follow.

Managing Topology Data

When you export topology data, you export only the existing topology into the data warehouse. The data is used in its entirety. It is not trimmed or reduced. Any previous topology data is replaced.

Run the `ovdwtopo` command to export the current topology data. The syntax of the `ovdwtopo` command for the export of topology data is:

```
ovdwtopo -export
```

Use the `-v` parameter to print out summary information of the actions taken.

Use the `-trace` parameter to capture additional information about the operation and store the trace in the file `$OV_LOG/ovdwtopo.trace` on UNIX and `*install_dir*\log\ovdwtopo.trace` on Windows operating systems.

Refer to the *ovdwtopo*(1) manpage or reference page for information about these and other options.

NOTE `ovtopmd` must be running for `ovdwtopo -export` to be successful.

Managing Trend Data

Trend data gathered by the SNMP data collector process, `snmpCollect`, is written to the SNMP binary database. When you export trend data, you may want to sum the data or trim to eliminate data that you do not need. Use `ovdwtrend` to manage trend data and delete unwanted binary data. Examples of the usage of `ovdwtrend` are shown below.

Run the `ovdwtrend` command to:

- Export some or all of the data to the data warehouse. The default export settings summarize trend data to one hour intervals.

  To export data reduced to one hour intervals, run:

  ```
  ovdwtrend -export
  ```

  To export raw data, run:

  ```
  ovdwtrend -exportto raw
  ```

  You need to make sure that the `snmpCollect` process has enough time to write all information to the binary database before you export it to the data warehouse. To accomplish this you should run the `ovdwtrend -export` no sooner than 30 minutes past the hour.

- Trim to eliminate data that is no longer needed in the data warehouse. If data is allowed to accumulate for a long period of time, the data warehouse will grow to an unmanageable size.

  To delete all data from the data warehouse, run:

  ```
  ovdwtrend -trim
  ```

  To delete data prior to a certain number of hours, such as data older than 48 hours, run:

  ```
  ovdwtrend -trimpriorto 48
  ```

  NNM daily performance reports for the web Reporting interface require that `snmp_reduced_trend` data reside in the data warehouse for at least a full day. If you have configured NNM performance reports and use the `ovdwtrend -trim` command, be sure to specify a `-trimpriorto` value of at least 48 hours.

- Summarize data in the data warehouse into daily, weekly, and monthly tables. The start and stop time for the summarization can be specified. For example, to summarize data collected between 8am and 5pm on weekdays, run the command:

```
ovdwtrend -sumstarttime 08:00 -sumstoptime 17:00 -sumnoweekends
```

- Delete numeric data in the SNMP database.

  To delete the exported data from the SNMP database that is older than two days, run:

  ```
  ovdwtrend -export -delpriorto 48
  ```

  (Source data deleted from the SNMP database does not appear in graphs generated with the graphing utility `xnmgraph`.)

To export trend data periodically, you can place the `ovdwtrend` command in `cron` (for UNIX systems) or a scheduler to run at specified times.

When data is saved in its reduced, not raw, form, some information is lost. The reduced data occupies less storage space, and practical information such as minimum and maximum values and the ability to compute mean and standard deviation is retained.

Refer to the *ovdwtrend*(1) manpage or reference page for information about these and other options.

NOTE The commands `ovcoltosql` and `ovcoldelsql` are still available in NNM. Their functionality has been incorporated into `ovdwtrend`. In future releases, `ovcoltosql` and `ovcoldelsql` may not be supported.

Storing Textual Data

When string data is exported to the data warehouse, only new values are stored. Repeat values are indicated by updating the `period` field to increase the length of time the value has been the same. String data is not aggregated into Daily, Weekly, Monthly, or Yearly tables.

When you export text data to the data warehouse, the data is deleted from the trend database. This is the only way to remove text data from the trend database.

The following SQL query returns one row for each `dataset_id` in the table, along with the timestamp and value of the most recent collection. Although this query is as efficient as possible in SQL, it does run slowly.

```
select txt.dataset_id,
 txt.sample_time,
 txt.value
from snmp_text_collections txt
 where txt.sample_timestamp =
 ( select max(sample_timestamp)
 from snmp_text_collections txt2
 where txt2.dataset_id = txt.dataset_id ) ;
```

To manage textual data in the data warehouse, `ovdwtrend` has two specific options, `-trimtextdays` and `-trimtextcount`, which affect only text data.

Table 6-1

ovdwtrend Option	Effect on Text Data	Effect on Numerical Data
`-export`	Exports and deletes from trend database.	Exports and leaves in trend database.
`-del*`	No effect.	deletes from trend database.
`-trimtext*`	Trims from data warehouse.	N/A.
`-trim*` (other than `-trimtext*`)	Trims from data warehouse.	Trims from data warehouse.

Exporting SNMP Trend Data to a Remote Relational Database Server

The data warehouse does not provide a distribution facility to combine SNMP trend data collected by multiple collection stations. As mentioned in “Distributing the Data Warehouse” on page 36, you can use the database networking facilities for accessing a database located on a different system than the management station or collection stations, and you can use one database server to store SNMP trend data collected from multiple collection stations. Additional information is available in the *Guide to Scalability and Distribution* and the Remote Database Connectivity white paper.

To avoid data consistency problems, consider the following issues:

- All collection stations must use the same DNS name server or else a given node may be represented as two different nodes in the trend database.
- All collection stations must use the same NLS Locale.
- Only SNMP trend data exports can be performed from different systems. You should perform the sum and trim operations (`ovdwtrend -sum -trim`), from one system, preferably only on a daily basis.
- The `ovdwtrend` tool can support multiple concurrent exports from different collection stations, however, you should schedule these exports to minimize concurrent exports.
- The SNMP Data Collectors on different collection stations should NOT attempt to collect the same SNMP data (node/MIB OID) from the same SNMP agents. While `ovdwtrend` does not allow duplicate data to be stored in the database, inconsistencies could arise from different collection periods, depending upon the sequence of the exports.
- If more than one station exports data for node X and you are doing joins against the topology tables (which should be exported only by the central management station), the joins against the topology tables should use `host_name`, not `topo_id`. `host_name` is interpreted by topology as the primary version of the object, while `topo_id` is interpreted by trend as being derived from whatever station last exported. Since `topo_id` is interpreted differently in the different data tables, it should not be used for the join.

The considerations described above apply only to SNMP trend data transferred to a central database server. The postmaster can send events to a central management station, and the topology manager can be configured to forward topology data to a central management station.

Managing Event Data

Event data is written to the event database. Since the event database collects large amounts of event data, you can reduce the data by either filtering the events or trimming the data. Use `ovdwevent` to manage event data.

Events that are filtered are not exported. You can create event filters by:

- Adding entries to the `nnm_event_limits` table by using the utility `ovdweventflt` (see “Managing Event Filters Using ovdweventflt” on page 113). This table is empty by default.
- Using the `ovdwevent` options `-exportnoweekends` or `-exportstart`/`-exportstop`.

You can trim event data by using the `trim` options of the `ovdwevent` utility.

Examples of the usage of `ovdwevent` are shown below. Run **`ovdwevent`** to:

- Export data from the event database. The export sends all events since the last export.

  The time frame for the events can be specified. For example, to export events that occurred between 8am and 5pm on weekdays, run the command:

```
ovdwevent -exportstart 08:00 -exportstop 17:00 -exportnoweekends
```

- Export data to a specific table. Exported data is placed in the `nnm_event_detail` table by default. There are also tables for daily, weekly, monthly, and yearly data.

  Every time the `ovdwevent` command is run, the event tables are updated. If the command is run for the first time during the week, both the daily and the weekly summarization tables are updated.
- Trim to eliminate data that is no longer needed in the data warehouse. If data accumulates for a long period of time, the data warehouse will grow to an unmanageable size.

  To delete data in the detail table that is older than 180 days and data in other tables that is more than 12 periods old, run:

  ```
  ovdwevent -trim
  ```

You can be more specific about which data should be deleted. For example, to trim all details that are older than 60 days and summarize any data that is older than 5 days, run:

```
ovdwevent -trimdetail 60 -trimaggregate 5
```

The `trim` parameter provides options:

- You can also delete specific data from the data warehouse, such as data in a specific table, with a specific node name, or with a specific MIB object identifier.
- You can view the number of trimmed events before trimming by using the `trimnodelete` option.

To collect event data periodically, place the `ovdwevent` command in `cron` (for UNIX systems) or a scheduler to run at specified times.

The `ovdwevent` command does not delete data from the event database.

Refer to the *ovdwevent*(1) manpage or reference page for information about these and other options.

Managing Event Filters Using ovdweventflt

`ovdweventflt` manages event filters that determine “which events” for “what nodes” will and will not be exported to the NNM data warehouse. NNM stores the filters in the `nnm_event_limits` table of the data warehouse.

`ovdweventflt` provides the capability to perform the following tasks:

- Inserting new event export filters into the `nnm_event_limits` table.
- Updating existing event export filters in the `nnm_event_limits` table.
- Deleting existing event export filters from the `nnm_event_limits` table.
- Displaying existing event export filters in the `nnm_event_limits` table.

`ovdweventflt` does not export the event data to the NNM data warehouse or create tables and schema definitions in the relational database. Refer to the *ovdwevent* (1) manpage or reference page for information on exporting the event data.

Inserting an Event Filter

The node name and MIB object identifier identify each filter. To insert a new filter into the `nnm_event_limits` table, you must specify the node name (`-n` option) and MIB object identifier (`-o` option). You can configure a filter to prevent export, allow export, or limit the number of daily exports for a particular node/MIB object combination. The `-i` and `-limit` options determine the export capability for a filter. To filter (NOT export) the node/MIB object combination, specify `Y` for the `-i` option. If you do not want to filter (ALWAYS export) the node/MIB object combination, specify `N` for the `-i` option.

For example, the following command will prevent the export of all events generated by the node `riogrand`:

```
ovdweventflt -n riogrand -o '*' -i Y -u user -password passwd
```

If you do not specify the `-i` option, the value defaults to NULL and. `ovdweventflt` will use the value that you specify for the `-limit` option to determine the export capability of the filter.

The value associated with the `-limit` option specifies the number of events that `oveventflt` will export on a daily basis for a particular node/MIB object combination.

For example, the following command will allow the export of up to 10 occurrences of event `1.2.3.4.5.6.7.8.9` per day:

```
ovdweventflt -n '*' -o 1.2.3.4.5.6.7.8.9  -limit 10 -u user -password passwd
```

If you do not specify the `-i` and `-limit` options when inserting a new filter, they will default to the values NULL and 0, respectively. The default values prevent the specified node name/MIB object combination from being exported.

Refer to the *ovdweventflt*(1) manpage or reference page for more information about these and other options.

NOTE NNM can identify nodes in the following ways: node name only, fully qualified node name, or IP address. When creating a new filter, you should use the same value that is stored in the `nnm_event_detail` table. To determine this value, use the `SELECT DISTINCT NODENAME FROM NNM_EVENT_DETAIL ORDER BY NODENAME` SQL command.

Updating an Event Filter

To update an existing filter, follow the same rules as creating a new filter. You must specify both the node name and MIB object identifier. The values for the `-i` and `-limit` options will be updated to the values you specify on the command line. If you do not specify `-i` or `-limit` when updating an existing filter, `ovdweventflt` uses the default values, NULL and 0, respectively.

Refer to the *ovdweventflt*(1) manpage or reference page for more information about these and other options.

Deleting an Event Filter

Deleting an event filter requires that you specify the node name and MIB object identifier. `ovdweventflt` does not allow you to specify the `-display`, `-i` and `-limit` options when deleting a filter.

For example, the following command will delete the filter used in the previous example:

```
ovdweventflt -n '*' -o '1.2.3.4.5.6.7.8.9' -delete -u user -password passwd
```

You can delete all of the filters by specifying the `-delete` and `-clean` options on the command line. `ovdweventflt` does not allow you to specify the `-n` and `-o` options when using the `-clean` option.

The following command will delete all filters in the `nnm_event_limits` table:

```
ovdweventflt -delete -clean -u user -password passwd
```

Refer to the *ovdweventflt*(1) manpage or reference page for more information about these and other options.

Displaying the Event Filters

Specify the `-display` option to display all of the filters in the `nnm_event_limits` table. `ovdweventflt` does not allow you to specify the `-delete` option with the `-display` option. To display event filters associated with a specific node and MIB object, you can specify a specific node name and MIB object identifier.

The following command displays all filters in the `nnm_event_limits` table:

```
ovdweventflt -display -u user -password password
```

The event filters are sorted by the MIB object identifier and displayed in the same order that `ovdwevent` applies the filters. The order that `ovdwevent` applies the filters is:

- Filters with explicit values for the node name and MIB object identifier fields.
- Filters with a wildcard for the node name and an explicit value for the MIB object identifier.
- Filters with an explicit node name and a wildcard for the MIB object identifier.
- Filters with wildcards for both the node name and MIB object identifier.

Refer to the *ovdweventflt*(1) manpage or reference page for more information about these and other options. See the *ovdwevent*(1) manpage or reference page for information on event filter priority.

NOTE When using the wildcard symbol (*) in the `ovdweventflt` command, you may need to enclose it within single quotes. This prevents your command shell from expanding it out to the names of every file in the current directory.

Using a Scheduler

If the web Reporting interface reports do not provide the information you need, you can schedule NNM to export data to the data warehouse reliably and regularly. Schedule exports to prevent overlapping exports of different data types.

- For UNIX systems, use the standard UNIX tool, `cron`.
- For Windows operating system, use a scheduling program.

NOTE To minimize database problems, specify export, sum, and trim operations on a single invocation of each of the data warehouse maintenance tools (`ovdwtrend`, `ovdwtopo`, and `ovdwevent`). The tools export, summarize, and then trim the data, if specified.

Scheduling Your Trend Exports to Coexist with the Web Reporting Interface

NNM automatically manages data coexistence for trend data by using different names for web Reporting interface generated data. However, problems can arise from conflicting export times of trend data and contention for access to the data warehouse database.

If your scheduled export is in operation at a given time, the web Reporting interface processes cannot export to the database at the same time. Conversely, while NNM is exporting to the data warehouse for report generation, your scheduled exports cannot access the database. Without exports, there is no data from which to report.

You can avoid blocking NNM by configuring your own exports either early enough to be completed before NNM's scheduled times, or after NNM's exports have completed. The web Report interface exports trend data hourly at 40 minutes past each hour. If your export blocks NNM, NNM will attempt to reschedule. See "Reviewing Report Schedules Using request_list" on page 222 for more information.

These concerns do not affect exports of event or topology information; only SNMP trend data is affected.

Explicitly Scheduling Export with cron (for UNIX Systems)

The following code excerpts provide examples for scheduling data warehouse exports if the web Reporting interface does not match your needs. To use the examples below to schedule data export:

1. Login as user **`root`**.
2. Use **`crontab -e`** to edit the `cron` entries.
3. Copy the relevant code into the root user's `crontab` file.
   (HP OpenView recommends that the comment lines be included in the `crontab` file to assist in the modification of these entries.)
4. Consult the HP OpenView Administrator to determine the correct settings for your installation.

See the *cron*(1M) and *crontab*(1) manpages for more information about `cron`.

Scheduling Export of Topology Data

To schedule a daily export of topology information:

```
# Schedule a daily export of HP OpenView Topology information.
# Schedule for 11:50pm every day.
# This program replaces existing Topology data in the Data Warehouse.
# You must use caution not to schedule this command near any backup
# processes you are also performing on this machine.
# See the ovdwtopo(1) man page for more information.
50 23 * * * /opt/OV/bin/ovdwtopo -export
```

Scheduling Export of Event Data

To schedule the export of event information:

```
# Schedule the export of HP OpenView Event information.
# This program adds new information the HP OpenView Event data store.
# Only use the 'trim' option once a day.
# Schedule for 15 minutes after the hours of 2,5,8,11,14,17 and 20 every day.
# You must use caution not to schedule this command near any backup
# processes you are also performing on this machine.
# See the ovdwevent(1) man page for more information.
15 23 * * * /opt/OV/bin/ovdwevent -export -trimdetail 14
15 2,5,8,11,14,17,20 * * * /opt/OV/bin/ovdwevent -export
```

Scheduling Export of Trend Data

To schedule the export of trend information:

```
# Schedule the export of HP OpenView Trend information.
# You must setup the capture of trend information using xnmcollect.
# This program adds new information the HP OpenView Event data store.
# Only use the 'trim' and 'sum' options once a day.
# Schedule for 35 minutes after the hours of 2,5,8,11,14,17 and 20 every day.
# You must use caution not to schedule this command near any backup
# processes you are also performing on this machine.
# See the ovdwtrend(1) man page for more information.
35 23 * * * /opt/OV/bin/ovdwtrend -export -sum -trim -trimpriorto 236
35 2,5,8,11,14,17,20 * * * /opt/OV/bin/ovdwtrend -export
```

7 Custom SQL Reporting

Using Schemas

The data warehouse consists of the four data schemas described below. Use the query commands described later in this chapter to access this data. See Chapter 8, “Accessing the Data Warehouse Schemas,” on page 131 for specific information about the schemas.

The data schemas are described in Table 7-1.

Table 7-1 Descriptions of the Data Warehouse Schema Tables

Table	Functions
`snmpCollect` data `tables_trend.[solid oracle msSqlSrvr]`	Reduce the collected data into hourly records for short term analysis.
	Aggregate the data to generate daily, weekly, and monthly representations for long term trend analysis.
Topology data `tables_topo.[solid oracle msSqlSrvr]`	Generate tables for nodes, segments, and interfaces.
	Create a capabilities table to classify, group, and order node data.
Event data `tables_event.[solid oracle msSqlSrvr]`	Filter out unwanted traps and events to create a complete record of desired events.
	Generate specialized tables to analyze threshold violations and node availability.
	Aggregate to summarize basic facts about exported events to enable longer term analysis.
Common data `tables_common.[solid oracle msSqlSrvr]`	Hold operational data such as time stamps for control of exports.

You can find the schema definitions in:

Windows: `install_dir\conf\analysis\sqlscripts`

UNIX: `$OV_CONF/analysis/sqlscripts`

NNM provides the schema definitions for writing reports. You should not modify the schemas.

NOTE The schemas are read-only. You can query the tables at any time for report generation, but you must not write to the tables. *Writing to the tables directly is not supported by Hewlett-Packard.*

Performing Queries on the Data Warehouse

Once the data is exported to the database tables, you can query the schema to extract the specific data you need to create reports. You can access the data in the data warehouse with the database vendor's own query tools.

The data warehouse provides the `ovdwquery` command to access data. `ovdwquery` is a simple, interactive, ODBC-enabled query tool, similar in concept to Oracle's SQLplus and iSQL with SQL*Plus. The `ovdwquery` command:

- Reads valid SQL statements from `stdin` (or the command prompt) or a file (specified with the `-file` option).
- Sends output to `stdout` (or the command prompt) or a file (specified with the `-out` option). Output data columns are separated by tabs unless another character such as white space, commas, or other character specified by the `-sep` option.

The SQL query can contain one or more statements. The query can target the NNM schema described in Chapter 8, "Accessing the Data Warehouse Schemas." See *ovdwquery*(1) for more information about the utility and its options.

You can also query the SNMP trend data using `ovcolqsql`. This utility retrieves summary data from the trend database and does not read SQL commands. See "Queries of the Trend Schema" on page 127 for more information about this utility.

In order to perform joins across trend, topology and event data tables, the data must reside in the same relational database. To facilitate these joins, the data warehouse only uses one relational database. If you want to migrate to another database, see Chapter 4, "Migrating NNM Data Between Databases."

Examples of SQL statements that NNM uses to query the three data types are included below. Use `ovdwquery`, `ovcolqsql`, or other database tools to send these SQL statements to the database and return the requested data.

Querying the Data Warehouse

You can send a query to the data warehouse by several different methods. Examples are shown below:

- To send a query as standard input using the default behaviors of `ovdwquery`, type **`ovdwquery`** to start interactive mode, then enter the SQL commands. SQL commands must be terminated with a semi-colon (;). Use **`quit;`** to terminate the `ovdwquery`.

```
ovdwquery -u user -password password
select * from snmp_trend_desc;
quit;
```

- To send a query by storing commands in a file that explicitly names the query file as `inventory.qry`, the output file as `inventory.out`, and produces verbose output to `inventory.err`:

```
ovdwquery -u user -password password -file inventory.qry -v > inventory.out 2>
inventory.err
```

- To send a query using `ovcolqsql` to produce summary data from the `snmp_reduced_trend` table for the MIB object `ifInOctets` on interface `4` of host `jane`:

```
ovcolqsql -h jane -m ifInOctets -n 4
```

`ovdwquery` terminates if the SQL statement has errors, for example, syntax errors or a referenced object does not exist. If you specify `ovdwquery` with the `-force` option, `ovdwquery` will not terminate when it encounters an error.

SQL Queries of the Topology Schema

You can access topology data stored in the data warehouse by using standard SQL query tools. You can find examples of SQL queries of topology data in the directory:

Windows:	`install_dir\contrib\NNM\topo`
UNIX:	`$OV_CONTRIB/NNM/topo`

Examples of SQL statements used to query the topology tables appear below.

The following example illustrates how to join the `nnm_nodes` and `nnm_interfaces` tables to get a list of nodes and their interfaces. The output of this example orders the results according to the nodename's IP hostname and the interface class SNMP interface name.

```
SELECT n.ip_hostname, i.snmp_ifname, i.ip_address, o.ip_status
FROM nnm_nodes n, nnm_interfaces i, nnm_objects o
WHERE i.node_id = o.ovw_id
AND o.topo_id = n.topo_id
ORDER BY n.ip_hostname, i.snmp_ifname;
```

The following example illustrates how to list all interfaces on a node.

```
SELECT snmp_ifdescr
FROM nnm_interfaces i, nnm_nodes n, nnm_objects o
WHERE n.ip_hostname = 'your_node_name.domain'
AND n.topo_id = o.topo_id
AND o.ovw_id = i.node_id;
```

The following example illustrates how to get information about nodes that are routers.

```
SELECT ip_hostname, snmp_sysdescr, snmp_syslocation, snmp_syscontact
FROM nnm_nodes n, nnm_nodes_cap c
WHERE n.topo_id = c.topo_id
AND c.isRouter = 1;
```

SQL Queries of the Event Schema

You can access event data stored in the data warehouse by using standard SQL query tools. You can find examples of SQL queries of event data in the directory:

Windows:	*`install_dir`*`\contrib\NNM\event`
UNIX:	`$OV_CONTRIB/NNM/event`

You cannot use some built-in values, such as `$BEGIN_TODAY` and `$NOW`, with standard query tools, but are acceptable input with `ovdwquery`. You can use the following built-in values with the SQL statement, `WHERE` clause:

- `$NOW`
- `$BEGIN_TODAY`
- `$BEGIN_WEEK` (Sunday, 00:00)
- `$BEGIN_MONTH`

- `$BEGIN_YEAR`
- `$BEGIN_YESTERDAY`
- `$BEGIN_LAST_WEEK` (Sunday, 00:00)
- `$BEGIN_LAST_MONTH`
- `$BEGIN_LAST_YEAR`

For example, the SQL statements below join the tables `nnm_event_cat` and `nnm_event_detail` to determine the number of events in a particular event category that have occurred since midnight:

```
SELECT c.text, count(event_uuid)
FROM nnm_event_cat c left outer join nnm_event_detail d
ON c.category = d.category
AND $BEGIN_TODAY >= d.event_timestamp
AND d.event_timestamp < $NOW
GROUP by c.text
ORDER by 1 desc;
```

Queries of the Trend Schema

You can access trend data stored in the data warehouse by using standard SQL query tools. You can find examples of SQL queries of trend data in the directory:

Windows:	`*install_dir*\contrib\NNM\trend`
UNIX:	`$OV_CONTRIB/NNM/trend`

The data warehouse provides a limited query tool for trend data, `ovcolqsql`. This tool displays summary data for the requested trend data sets to `stdout`. The following summary data is provided for each data set:

- Sample time
- Period
- Minimum value
- Maximum value
- Average
- Standard deviation

You can use `ovcolqsql` to display detailed data by using the `-d raw` option. If this option is chosen, both summary data and each record in the specified dataset are displayed. Refer to the *ovcolqsql*(1) manpage or reference page for information about other options available when using this command.

You can also use `ovdwquery` to send SQL statements to the trend tables.

The following SQL statements report Cisco router performance and make daily summaries of the results ordered by hostname, title, and instance.

```
SELECT  snmp_trend_dataset.host_name,
snmp_trend_dataset.instance,
snmp_trend_desc.title,
ROUND(AVG(snmp_reduced_trend.data_sum/snmp_reduced_trend.data_count),3)
FROM    snmp_trend_desc, snmp_reduced_trend, snmp_trend_dataset
WHERE   snmp_trend_desc.descript_id = snmp_trend_dataset.descript_id
AND   snmp_reduced_trend.dataset_id = snmp_trend_dataset.dataset_id
AND   snmp_reduced_trend.sample_timestamp BETWEEN ($BEGIN_TODAY) AND ($NOW)
AND   (   snmp_trend_desc.title = 'NNM_CiscoSysUptime'
OR snmp_trend_desc.title = 'NNM_CiscoFreeMem'
OR snmp_trend_desc.title = 'NNM_Cisco1MinAvgBusy'
OR snmp_trend_desc.title = 'NNM_Cisco5MinAvgBusy'
OR snmp_trend_desc.title = 'NNM_CiscoNowBusy' )
GROUP BY snmp_trend_dataset.host_name,
snmp_trend_dataset.instance,
snmp_trend_desc.title
ORDER BY snmp_trend_dataset.host_name,
snmp_trend_dataset.instance,
snmp_trend_desc.title;
```

Using SQL Statements for Reporting

Writing SQL statements that allow you to retrieve the desired data from the data warehouse facilitates useful reporting. The following examples of SQL statements provide the data to generate availability, exception, and inventory reports.

The following example retrieves all threshold events for today until the present time:

```
SELECT distinct d.nodename,d.event_time,s.text,d.message
FROM nnm_event_detail d, nnm_event_sev s, nnm_event_cat c
WHERE $BEGIN_TODAY >= event_timestamp
AND event_timestamp < $NOW
AND c.text = 'Threshold'
AND c.category=d.category
AND d.severity=s.severity;
```

The following example retrieves all routers that exist on the network 192.168.55 with some associated information:

```
SELECT    n.ip_hostname, i.ip_address, i.ifnumber,
n.snmp_sysobjid, c.vendor_name, c.agent_name
FROM     nnm_nodes n, nnm_interfaces i, nnm_objects o, nnm_objects netobj,
nnm_networks nw, nnm_nodes_cap c
WHERE    c.isRouter = 1
AND      n.topo_id = c.topo_id
AND      n.topo_id = o.topo_id
AND      o.ovw_id = i.node_id
AND      i.ip_address <> '0.0.0.0'
AND      i.net_id = netobj.ovw_id
AND      nw.topo_id = netobj.topo_id
AND      nw.ip_network_name = '192.168.55'
ORDER BY n.ip_hostname, i.ip_address;
```

8 Accessing the Data Warehouse Schemas

The HP OpenView schema include topology data tables, SNMP MIB trend data tables, event data tables, and common data tables.

This chapter provides information about the definitions and contents of these tables. You can generate reports or perform queries via any report tools supported by the database in use as described in Figure 7, “Custom SQL Reporting.”

Introduction

When you use NNM to manage your network, you generate data. This data is stored in three separate NNM databases. To use this data for report generation, you must first export the data to the data warehouse, using commands, `ovdwtopo`, `ovdwevent`, or `ovdwtrend` as described in Chapter 6, "Export Utilities."

When the data is exported to the data warehouse, it is stored in tables. From these tables, you are able to query the data and format it in reports, using the tools as described in Chapter 7, "Custom SQL Reporting," on page 121.

The specific structure, collection of tables, and relationship rules among data is called a schema. In the data warehouse environment, there are four schemas as described below. The common schema contains data from all three of the data types. The data warehouse schemas include:

- Topology data schema

  Topology data stores information about nodes, networks, and the core network topology.

  Use the `ovdwtopo` utility to export the entire topology database from the topology database to the data warehouse.

- SNMP collected trend data schema

  SNMP collected data is obtained from SNMP agents.

  Use the `ovdwtrend` utility to export trend data from the `snmpCollect` binary database to the data warehouse.

- Event data schema

  Event data is obtained from the postmaster and is stored in the event database.

  Use the `ovdwevent` utility to export event data from the event database to the data warehouse.

- Common data schema

  This schema consists of operational data, such as time stamps for use in controlling your export operations.

NOTE All of the tables defined in this chapter are intended to be read-only. You can query the tables at any time for report generation, but you must not write to the tables. *Writing to the tables directly is not supported by Hewlett-Packard*.

The contents of these tables may change in future releases, but every attempt will be made to provide a migration path.

Entity Relationship Diagrams

The relational database is able to join tables together to obtain more information about a specified field. Entity diagrams of the schema described in the following tables may help you determine how to join the data tables together. A separate entity diagram is provided for each data type:

- For topology data, see Figure 8-1.
- For trend data, see Figure 8-2.
- For event data, see Figure 8-3.

Figure 8-1 Entity Diagram for the Topology Data Tables

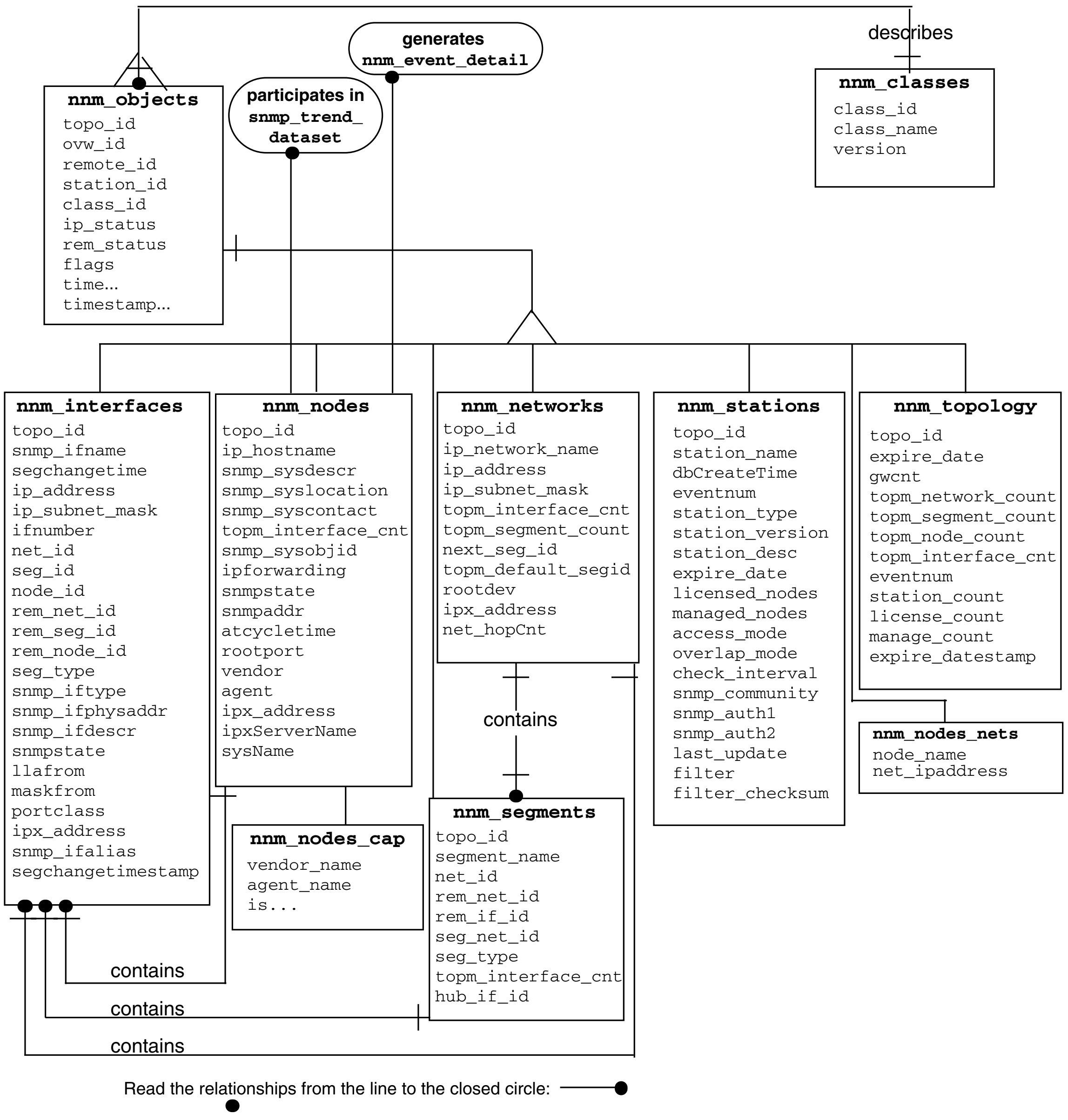


Figure 8-2 Entity Diagram for the Trend Data Tables

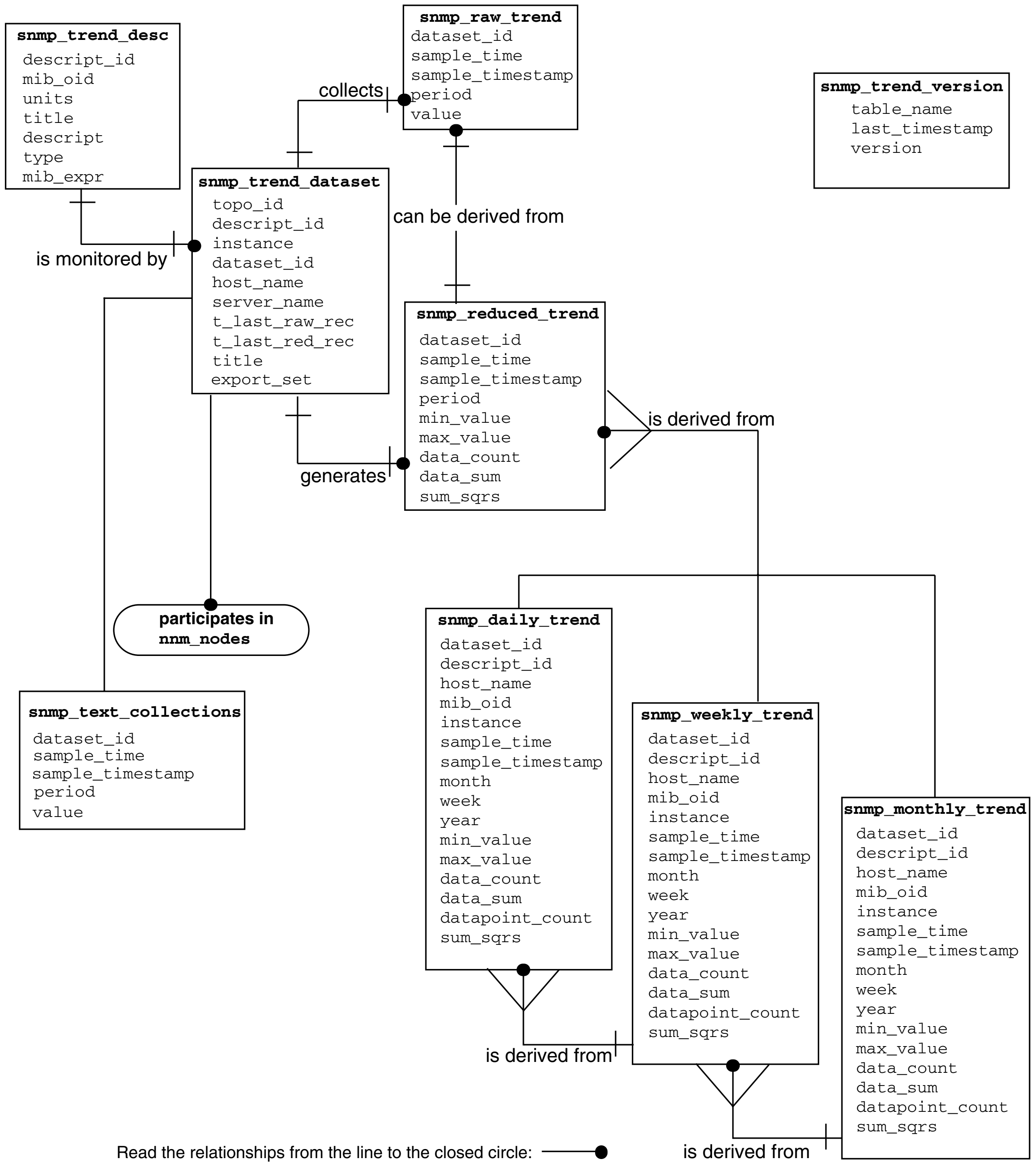
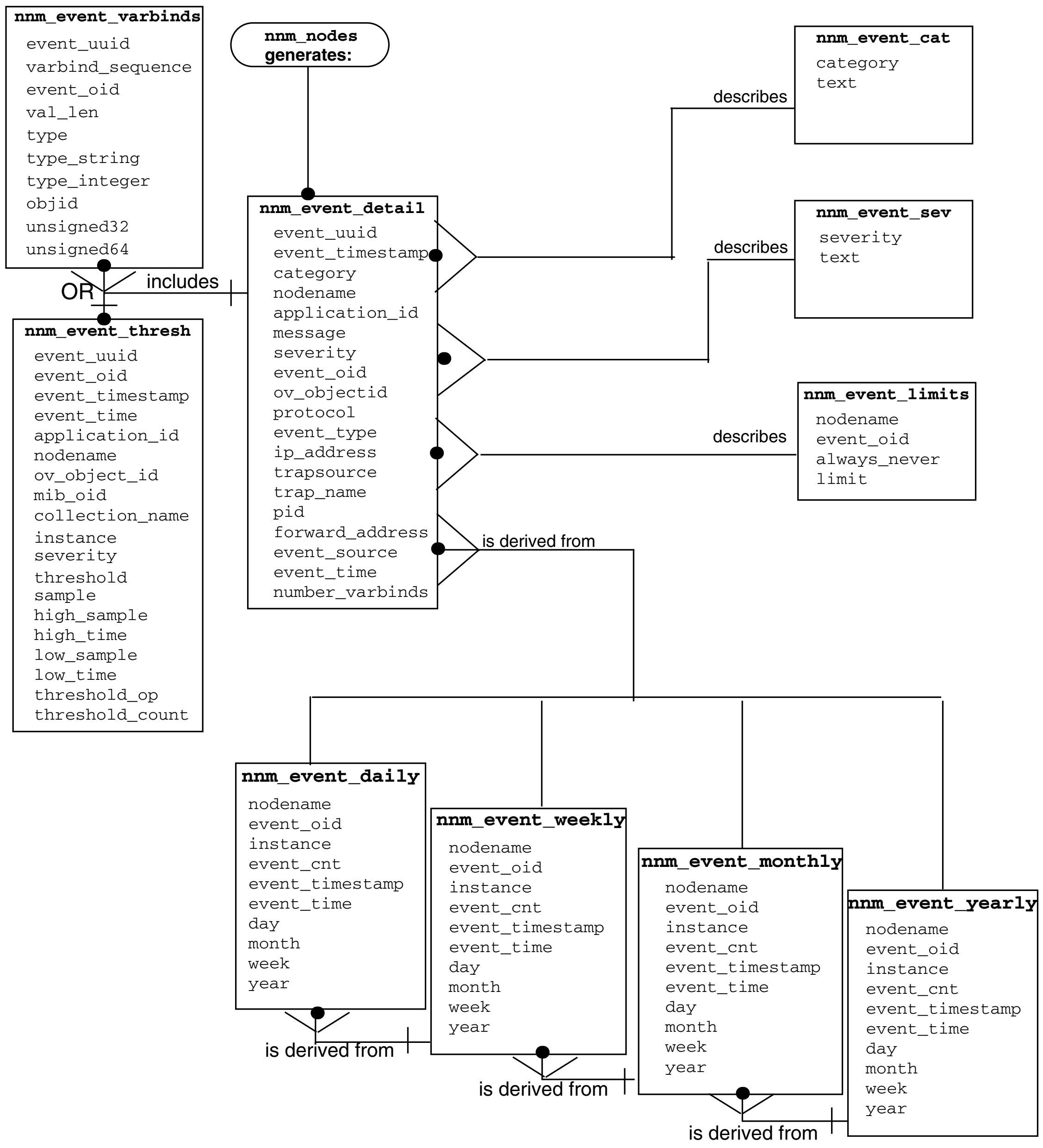


Figure 8-3 Entity Diagram for the Event Data Tables



Read the relationships from the line to the closed circle: ——●

General Conventions

Fields that are primary keys are also NOT NULL, UNIQUE, and INDEXES.

Fields labeled `_time` are stored in the date/time format: *`yyyy-mm-dd hh:mi:sec`*.

Fields labeled `_timestamp` are expressed in Epoch seconds, the number of seconds since January 1, 1970.

NNM's data schemas use the following conventions:

- Object IDs: The object IDs (such as the `seg_id` topology attribute) are 36-byte character Universal Unique Identifiers (UUIDs).
- Some data is converted before the data is placed into the tables. Table 8-1 lists the converted data fields.

Table 8-1 Conversions of Topology Data Fields

Data Field	Conversion
Time	IP topology values which represent time are converted to the database's internal date format before being stored.
IP Addresses	IP addresses/masks stored by `ovtopmd` are converted to a character string in standard dot notation for storage in the data warehouse.
Physical Addresses	Physical addresses stored by `ovtopmd` are converted to a character string for storage in the data warehouse. The character string representing the physical address is 0x followed by a string of hexadecimal digits.
SNMP OID	SNMP object IDs stored by `ovtopmd` are converted to character strings for storage in the data warehouse.
Unsigned Integers	Since some databases do not support unsigned integers, `ovtopmd` unsigned integers may be stored as signed integers within a database. In large networks, some count fields may contain unsigned values that are displayed as negative numbers via database reporting tools. These count fields are identified in the NNM tables with an asterisk (*).

Language Conventions

Some fields in the schema contain messages that are language-dependent. The values for these fields differ for different language environment variables, `LANG`. To get message fields to display in the same language as NNM is running, use the same `LANG` environment variable when you start NNM and when you export data (for example, `ovdwevent -export`).

Language-dependent fields are specified by a “*” in the Data Type column of the schema tables.

Topology Data Schema

The topology object model consists of the following classes of objects:

- `nnm_objects`
- `nnm_topology`
- `nnm_networks`
- `nnm_segments`
- `nnm_nodes`
- `nnm_interfaces`
- `nnm_stations`
- `nnm_classes`
- `nnm_nodes_cap`
- `nnm_nodes_net`

The last six objects in this list are derived from the first table, a generic topology object table called `nnm_objects`.

Each object class corresponds to a table in the topology schema. Information common to all topology objects is stored in `nnm_objects`. The topology tables share the same key, so you can access an object's complete definition by joining the generic object table to a specific class table via the `topo_id`.

In addition to the seven object tables, this schema also has a class table, `nnm_classes`, which contains an entry for each of the seven classes. Each entry in this class table contains the class name, id, and schema version number.

The topology database contains all of the topology information. When the topology database is exported to the data warehouse via the `ovdwtopo` utility, the topology schema tables are created. The `ovdwtopo` utility creates the following schemas, depending on which NNM-supported database you are using. The table definitions are found in the directory:

Windows:	`*install_dir*\conf\analysis\sqlScripts`
UNIX:	`$OV_CONF/analysis/sqlScripts`

The files containing the table information are:

- For the Oracle database: `tables_topo.oracle`
- For the NNM embedded database: `tables_topo.solid`
- For Microsoft SQL Server: `tables_topo.msSqlSrvr`

Once the data is in the topology tables shown on the following pages, the data can be used to generate reports.

NNM Objects Table (nnm_objects)

The `nnm_objects` table contains the fields common to all topology objects. Every version of every object has an entry in this table. In object-oriented terms, this table forms the base class for the station, network, segment, node, interface, and global topology objects.

Table 8-2 nnm_objects Topology Table

Attribute Name	Data Type	Constraint	Description
`topo_id`	VARCHAR	PRIMARY KEY	The Universal Unique Identifier (UUID) uniquely identifies each entry. Every object has a unique UUID that can be used to match the entries in different tables.
`ovw_id`	INTEGER	NOT NULL INDEX	The OVW object ID for the object. Multiple versions of the same type of object may have the same OVW ID when an object is monitored in a distributed environment by multiple management stations.
`remote_id`	INTEGER	NOT NULL	The OVW Object ID of the object as reported by the remote station monitoring this version of the object. For locally-monitored objects, the `remote_id` is the same as the `ovw_id`.
`station_id`	INTEGER	NOT NULL	The local OVW Object ID of the station monitoring this version of the object, or zero (0) for locally-managed objects.
`class_id`	VARCHAR		Identifies whether the object is a station, network, segment, node, interface, or the global internet object. Each type of object has a class entry in the `nnm_classes` table.
`ctime`	VARCHAR	NOT NULL	The creation time of the object (the first time the object was placed in the database).
`mtime`	VARCHAR	NOT NULL	The last modification time of the object.

Table 8-2 nnm_objects Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`sym_time`	VARCHAR	NOT NULL	The last time the object was modified in a way that would affect the symbol for the object on a map.
`stat_time`	VARCHAR	NOT NULL	The last time the status of the object changed.
`label_time`	VARCHAR	NOT NULL	The last time the object was modified in a way that would affect the label for the object on a map.
`flags_time`	VARCHAR	NOT NULL	The last time the internal flag for the object changed.
`ip_status`	INTEGER	NOT NULL	The IP-derived status of the object. The valid status values are the same as those accepted by ovw (0: No status; 1: Unknown; 2: Normal; 3: Minor; 4: Critical; 5: Unmanage; 6: Warning; 7: Major; 8: Restricted; 9: Testing; 10: Disabled).
`rem_status`	INTEGER	NOT NULL	The IP status of the object, as reported by the station monitoring the object (0: No status; 1: Unknown; 2: Normal; 3: Minor; 4: Critical; 5: Unmanage; 6: Warning; 7: Major; 8: Restricted; 9: Testing; 10: Disabled).
`flags`	INTEGER	NOT NULL	A field used to store internal-only, Boolean-valued flags.
`extra0`	INTEGER	NOT NULL	(Reserved for future use.)
`extra1`	INTEGER	NOT NULL	(Reserved for future use.)

Table 8-2 nnm_objects Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
ctimestamp	INTEGER	NOT NULL	The first time the object was stored in the topology database expressed in Epoch time.
mtimestamp	INTEGER	NOT NULL	The last time the object was stored in the topology database expressed in Epoch time.
sym_timestamp	INTEGER	NOT NULL	The time of the last change affecting the object's symbol expressed in Epoch time.
stat_timestamp	INTEGER	NOT NULL	The time of the last change affecting the object's status expressed in Epoch time.
label_ timestamp	INTEGER	NOT NULL	The time of the last change affecting the object's label expressed in Epoch time.
flags_ timestamp	INTEGER	NOT NULL	The time of the last change affecting the object's flag expressed in Epoch time.

NNM Classes Table (nnm_classes)

The `nnm_classes` table maps a “class name” to a “class id” for use in referencing other tables and determining the versions of supported tables. Each class has a separate version so that, if a particular table needs to change for a particular object, the change does not impact other objects or tables.

Table 8-3 nnm_classes Topology Table

Attribute Name	Data Type	Constraint	Description
class_id	VARCHAR	PRIMARY KEY	The Universal Unique Identifier (UUID) of the class.

Table 8-3 nnm_classes Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`class_name`	VARCHAR	UNIQUE NOT NULL	The name of the class.
`version`	INTEGER	NOT NULL	The version of the table representing the class.

NNM Interfaces Table (nnm_interfaces)

The `nnm_interfaces` table includes all the interfaces contained in the topology. Each interface also has an entry in the `nnm_objects` table that can be cross-referenced via the `topo_id` field. Where a field is described as "The SNMP *item*," it refers to a field that is retrieved via SNMP during the discovery and monitoring process.

Table 8-4 nnm_ interfaces Topology Table

Attribute Name	Data Type	Constraint	Description
`topo_id`	VARCHAR	PRIMARY KEY	The Universal Unique Identifier (UUID) of the interface.
`snmp_ifname`	VARCHAR		The SNMP `ifName` (interface name).
`segchangetime`	VARCHAR	NOT NULL	The last time the interface changed segments.
`ip_address`	VARCHAR	INDEX	The IP address in dot notation.
`ip_subnet_mask`	VARCHAR		The IP subnet mask in dot notation.
`ifnumber`	INTEGER	NOT NULL	The SNMP `ifindex`. If the device cannot be contacted via SNMP, the value is 0.
`net_id`	INTEGER	NOT NULL INDEX	The OVW Object ID of the network containing this interface.

Table 8-4 nnm_ interfaces Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
seg_id	INTEGER	NOT NULL	The OVW Object ID of the segment containing this interface. May be NULL or 0, to indicate an unconnected interface.
node_id	INTEGER	NOT NULL INDEX	The OVW Object ID of the node containing this interface.
rem_net_id	INTEGER	NOT NULL	The OVW Object ID of the network containing this interface, as reported by the station monitoring this interface. For locally-monitored interfaces, the rem_net_id matches the net_id field.
rem_seg_id	INTEGER	NOT NULL	The OVW Object ID of the segment containing this interface, as reported by the station monitoring this interface. For locally-monitored interfaces, the rem_seg_id matches the seg_id field.
rem_node_id	INTEGER	NOT NULL	The OVW Object ID of the node containing this interface, as reported by the station monitoring this interface. For locally-monitored interfaces, the rem_node_id matches the node_id field.

Table 8-4 nnm_ interfaces Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`seg_type`	INTEGER	NOT NULL	An enumerated integer that indicates the segment type expected for this interface (as determined by the IP discovery process). Possible values are: 0 = Unset (indicates no preferred type) 1 = Bus Segment 2 = Star Segment 3 = Token Ring 4 = FDDI Ring 5 = Serial
`snmp_iftype`	INTEGER	NOT NULL	The SNMP `iftype` (interface type) according to the IANA definitions.
`snmp_ ifphysaddr`	VARCHAR		The SNMP `ifPhysaddr`, if the node supports SNMP; otherwise, this value is the physical address as determined by the discovery process.
`snmp_ifdescr`	VARCHAR		The SNMP `ifDescr` (interface description).
`snmpstate`	INTEGER		Used to store internal-only, Boolean-valued flags.
`llafrom`	INTEGER		The IP address of the node that the `snmp_ifphysaddr` was discovered from, or 0 (if the node itself).

Table 8-4 nnm_ interfaces Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`maskfrom`	INTEGER		An enumerated value indicating the source of the IP subnet mask. The possible values are: 0 = Unknown 1 = SNMP IP Address table 2 = A guess based on the containing network 3 = A guess based on the network class 4 = ICMP Mask request 5 = A guess based on user input
`portclass`	INTEGER		For hub and bridge interfaces, the port group for this interface.
`ipx_address`	VARCHAR		The IPX address.
`snmp_ifalias`	VARCHAR		The value of the object's interface. The value set for the device by the administrator (for example, an alias name that is more meaningful to the administrator). If no value is set, the default is `ifnumber`. Use this to join to the `instance` column of the SNMP tables.
`segchange timestamp`	INTEGER	NOT NULL	The last time the segment associated with the interface changed expressed in Epoch time.

NNM Networks Table (nnm_networks)

The nnm_networks table contains an entry for each network and subnet contained in the topology. Each network also has an entry in the nnm_objects table that can be cross-referenced via the topo_id.

Table 8-5 nnm_networks Topology Table

Attribute Name	Data Type	Constraint	Description
topo_id	VARCHAR	PRIMARY KEY	The Universal Unique Identifier (UUID) of the network.
ip_network_name	VARCHAR	NOT NULL INDEX	Either the IP Address of the network or the name of the network from the /etc/networks (or equivalent) configuration file.
ip_address	VARCHAR		The network IP address.
ip_subnet_mask	VARCHAR		The network IP subnet mask.
topm_interface_cnt	INTEGER	NOT NULL	The number of interfaces on this network.
topm_segment_count	INTEGER	NOT NULL	The number of segments in this network.
next_seg_id	INTEGER	NOT NULL	An internal counter used to generate unique segment names for segments in this network.
topm_default_segid	INTEGER	NOT NULL	The OVW Object ID of the default segment for this network. This value is the segment in which newly-discovered nodes are placed when they are discovered by netmon.
rootdev	VARCHAR		The IP address of the root device used in the IP discovery and layout processes.
ipx_address	VARCHAR		The IPX address.
net_hopCnt	INTEGER		The network hop count.

NNM Nodes (nnm_nodes)

The `nnm_nodes` table contains an entry for each node contained in the topology. Each node also has an entry in the `nnm_objects` table that can be cross-referenced via the `topo_id`.

Table 8-6 nnm_nodes Topology Table

Attribute Name	Data Type	Constraint	Description
`topo_id`	VARCHAR	PRIMARY KEY NOT NULL	The Universal Unique Identifier (UUID) of the node.
`ip_hostname`	VARCHAR	NOT NULL INDEX	The name of the node: either the IP address of one of the interfaces on the node or the name of the node as determined via IP hostname resolution of one of the interfaces.
`snmp_sysdescr`	VARCHAR		The SNMP `sysDescr` field. `netmon` obtains this information from the agent on the node, if possible.
`snmp_syslocation`	VARCHAR		The SNMP `sysLocation` field. `netmon` obtains this information from the agent on the node, if possible.
`snmp_syscontact`	VARCHAR		The SNMP `sysContact` field. `netmon` obtains this information from the agent on the node, if possible.
`topm_interface_cnt`	INTEGER	NOT NULL	The count of contained interfaces.

Table 8-6 nnm_nodes Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`snmp_sysobjid`	VARCHAR		The SNMP `sysObjectId` field. `netmon` obtains this information from the agent on the node, if possible. The `sysObjectId` may be used to determine the vendor and `snmpAgent` fields, various topology attributes, and the symbol for the node. See the *oid_to_type*(4) and *oid_to_sym*(4) manpages or reference pages.
`ipforwarding`	INTEGER	NOT NULL	The SNMP forwarding field. `netmon` obtains this information from the agent on the node, if possible.
`snmpstate`	INTEGER	NOT NULL	Used to store internal-only, Boolean-valued flags.
`snmpaddr`	VARCHAR		The IP address used for sending SNMP requests to the node. This field may change based on the status of various interfaces on the node.
`atcycletime`	INTEGER	NOT NULL	Time between discovery polls for this node. This time interval is dynamically-adjusted by `netmon`, based on the number of successful discoveries in the last discovery poll.
`rootport`	INTEGER	NOT NULL	Used with the IP discovery and layout processes to determine the bridge or hub topology.

Table 8-6 nnm_nodes Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`vendor`	INTEGER	NOT NULL	The vendor of the node based on the mapping established via the *oid_to_type* file. This enumeration matches the enumerated values in `ovwdb` for the "vendor" field.
`agent`	INTEGER	NOT NULL	The SNMP Agent of the node based on the mapping established via the *oid_to_type* file. This enumeration matches the enumerated values in `ovwdb` for the "SNMPAgent" field.
`ipx_address`	VARCHAR		The IPX address.
`ipxServerName`	VARCHAR		The IPX server name.
`sysName`	VARCHAR		The SNMP system name.

NNM Segments Table (nnm_segments)

The `nnm_segments` table contains an entry for each segment contained in the topology. Each segment also has an entry in the `nnm_objects` table that can be cross-referenced via the `topo_id`.

Table 8-7 nnm_segments Topology Table

Attribute Name	Data Type	Constraint	Description
`topo_id`	VARCHAR	PRIMARY KEY NOT NULL	The Universal Unique Identifier (UUID) of the segment.
`segment_name`	VARCHAR	NOT NULL INDEX	The name of the segment.
`net_id`	INTEGER	NOT NULL	The OVW Object ID of the network containing this segment.

Table 8-7 nnm_segments Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
rem_net_id	INTEGER	NOT NULL	The OVW Object ID of the network containing this segment as reported by a remote collection station. For locally-monitored segments, the rem_net_id matches the net_id.
rem_if_id	INTEGER	NOT NULL	For star-type segments, the OVW Object ID of the main hub of this segment, as reported by a remote collection station. For locally-monitored segments, the hub_if_id should match the rem_if_id.
seg_net_id	INTEGER	NOT NULL	The internal ID of this segment within this network, as based on the next_seg_id field in the network at the time the segment is created.
seg_type	INTEGER	NOT NULL	An enumerated value indicating the type of the segment. Possible values are: 0 = Unknown or unset 1 = Bus Segment 2 = Star Segment 3 = Token Ring 4 = FDDI Ring 5 = Serial
topm_interface_cnt	INTEGER	NOT NULL	The count of the number of interfaces contained in this segment.

Table 8-7 nnm_segments Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`hub_if_id`	INTEGER	NOT NULL	For star-type segments, the OVW Object ID of the main hub of this segment.

NNM Stations Table (nnm_stations)

The nnm_stations table contains an entry for each station contained in the topology. Each station also has an entry in the nnm_objects table that can be cross-referenced via the topo_id. Stations represent management stations that are being used as sources of topology information in a distributed environment. The management station also has an entry as the local source of topology via the topology and discovery processes.

Table 8-8 nnm_stations Topology Table

Attribute Name	Data Type	Constraint	Description
topo_id	VARCHAR	PRIMARY KEY NOT NULL	The Universal Unique Identifier (UUID) of the station.
station_name	VARCHAR	NOT NULL INDEX	The IP hostname of the station, determined in the same manner as the ip_hostname field for nodes.
dbCreateTime	INTEGER	NOT NULL	The time of topology database creation on the remote station. Used by ovrepld to determine when a synchronization may need to occur.
eventnum	INTEGER	NOT NULL	The event sequence number of the last event reported by the remote station. The event sequence number is used by ovrepld to detect lost events from a remote collection station.
station_type	INTEGER	NOT NULL	The type of the collection station. 0 = Unknown or unset 1 = NNM/UX 2 = OpenView for Windows 3 = NNM/NT

Table 8-8 nnm_stations Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`station_ version`	INTEGER	NOT NULL	The version of the topology data exported by the remote collection station.
`station_desc`	VARCHAR		A string description of the remote station.
`expire_date`	INTEGER		The date the license expires for the remote station.
`licensed_nodes`	INTEGER	NOT NULL	The number of nodes licensed that are monitored by the remote station.
`managed_nodes`	INTEGER	NOT NULL	The number of nodes currently managed by the remote station.
`access_mode`	INTEGER	NOT NULL	The type of access granted to the remote station: 1 = read only 2 = read/write Currently, all remote stations have a value of “read only,” and only the local station has a value of “read/write.”
`overlap_mode`	INTEGER	NOT NULL	The mode determining how to handle overlaps between this station and another station. Possible modes are: 0 = Allow overlap 1 = Delete 2 = Unmanage The `overlap_mode` only applies to the local management station.

Table 8-8 nnm_stations Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
check_interval	INTEGER	NOT NULL	How often ovrepld should verify the status of a managed remote collection station (in seconds).
snmp_community	VARCHAR		(Reserved for future use.)
snmp_auth1	VARCHAR		(Reserved for future use.)
snmp_auth2	VARCHAR		(Reserved for future use.)
last_update	INTEGER	NOT NULL	The time of the last event or object modification on the remote collection station (according to the remote).
filter	VARCHAR		The failover filter.
filter_ checksum	INTEGER	NOT NULL	(Reserved for future use.)

NNM Topology Table (nnm_topology)

The `nnm_topology` table contains the global topology information. Each table has only a single entry.

Table 8-9 nnm_topology Topology Table

Attribute Name	Data Type	Constraint	Description
`topo_id`	VARCHAR	PRIMARY KEY	The Universal Unique Identifier (UUID) of the global topology object.
`expire_date`	VARCHAR		The date the local license expires.
`gwcnt`	INTEGER	NOT NULL	The number of gateways in the topology.
`topm_network_count`	INTEGER	NOT NULL	The number of networks in the topology.
`topm_segment_count`	INTEGER	NOT NULL	The number of segments in the topology.
`topm_node_count`	INTEGER	NOT NULL	The number of nodes in the topology.
`topm_interface_cnt`	INTEGER	NOT NULL	The number of interfaces in the topology.
`eventnum`	INTEGER	NOT NULL	The event sequence number. Used when forwarding events from this management station to another as part of distributed monitoring.
`station_count`	INTEGER	NOT NULL	The number of stations in the topology.
`license_count`	INTEGER	NOT NULL	The number of nodes licensed to be monitored by this management station. A value of -1 represents an unlimited node license
`manage_count`	INTEGER	NOT NULL	The number of nodes currently monitored by this management station.

Table 8-9 nnm_topology Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
expire_ datestamp	INTEGER		Expiration date of the license.

NNM Nodes Capabilities Table (nnm_nodes_cap)

The nnm_nodes_cap table contains characteristics about the node itself. For example, the node might be a device that is a bridge, router, and workstation. These device functions are specified by this table. For every entry in the nnm_nodes table, there is a corresponding entry in the nnm_nodes_cap table. The integer values in the nnm_nodes_cap table are 0 for not true and 1 for true. A value of 1 indicates that the device is the described device.

Table 8-10 nnm_nodes_cap Topology Table

Attribute Name	Data Type	Constraint	Description
topo_id	VARCHAR	PRIMARY KEY NOT NULL INDEX	The Universal Unique Identifier (UUID) of the global topology object.
vendor_name	VARCHAR		Name of the manufacturer of the node device.
agent_name	VARCHAR		SNMP agent type. Can represent the operating system.
isComputer	INTEGER		The device is a computer.
isPC	INTEGER		The device is a PC.
isWorkstation	INTEGER		The device is a workstation.
isMini	INTEGER		The device is a mini computer.
isMainFrame	INTEGER		The device is a mainframe computer.
isConnector	INTEGER		The device is a connector.

Table 8-10 nnm_nodes_cap Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`isBridge`	INTEGER		The device is a bridge.
`isRouter`	INTEGER		The device is a router.
`isIPRouter`	INTEGER		The device is an IP router.
`isIPXRouter`	INTEGER		The device is an IPX router.
`isRepeater`	INTEGER		The device is a repeater.
`isHub`	INTEGER		The device is a hub.
`isSwitch`	INTEGER		The device is a switch.
`isDevice`	INTEGER		The device is a device.
`isPrinter`	INTEGER		The device is a printer.
`isAnalyzer`	INTEGER		The device is an analyzer.
`isIP`	INTEGER		The device is a node that supports Internet Protocol.
`isIPX`	INTEGER		The device is a node that supports IPX.
`isSNMP Supported`	INTEGER		The device supports SNMP.
`isSNMPProxied`	INTEGER		The device proxies SNMP.
`isNetWare Server`	INTEGER		The device is a netware server.
`isMcCluster Member`	INTEGER		The device is a node that `netmon` determines to be a member of an HP-UX Mc Cluster (for example, a Service Guard cluster).
`isCollection StationNode`	INTEGER		The device is an NNM station that provides information to another NNM station about the object that it is managing.

Table 8-10 nnm_nodes_cap Topology Table (Continued)

Attribute Name	Data Type	Constraint	Description
`isRDMI Supported`	INTEGER		The device supports remote DMI.
`isHTTP Supported`	INTEGER		The device supports HTTP.
`isHTTPManaged`	INTEGER		The device manages HTTP.
`isRMON`	INTEGER		The device implements the RMON MIB.
`isRMON2`	INTEGER		The device implements the RMON2 MIB.
`isDS1`	INTEGER		The device implements the DS1 MIB.
`isDS3`	INTEGER		The device implements the DS3 MIB.
`isFrameRelay`	INTEGER		The device implements the FrameRelay MIB.
`isSONET`	INTEGER		The device implements the SONET MIB.
`isATM`	INTEGER		The device implements the ATM MIB.
`isCDP`	INTEGER		The device implements the CDP MIB.

NNM Nodes Network Table

This table matches a specific node to a network. The `nnm_nodes_nets` table provides a way to see all nodes on a network without going through the `nnm_objects` and `nnm_interfaces` tables.

Table 8-11 nnm_nodes_nets Table

Attribute Name	Data Type	Constraint	Description
`node_name`	VARCHAR	NOT NULL	The name of the node.
`net_ipaddress`	VARCHAR	NOT NULL	The network IP address

SNMP Collected Data (Trend Data) Schema

The trend schema consists of the following tables:

- SNMP trend version table (`snmp_trend_version`)

  Provides the version numbers to confirm that the schema did not change version since the last time the tools were used.

- SNMP trend description table (`snmp_trend_desc`)

  Provides a description of the monitored MIB variables and MIB expressions.

- SNMP trend dataset table (`snmp_trend_dataset`)

  Defines a trend dataset by identifying the target host, MIB variable or expression, and instance.

- SNMP raw trend data table (`snmp_raw_trend`)

  Stores raw data collected from the SNMP environment.

- SNMP reduced trend data table (`snmp_reduced_trend`)

  Stores reduced data derived from raw SNMP data.

- SNMP aggregated daily trend data table (`snmp_daily_trend`)

  Stores aggregated daily trend data derived from raw SNMP data.

- SNMP aggregated weekly trend data table (`snmp_weekly_trend`)

  Stores aggregated weekly trend data derived from raw SNMP data.

- SNMP reduced trend data table (`snmp_monthly_trend`)

  Stores aggregated monthly trend data derived from raw SNMP data.

- SNMP text collections (`snmp_text_collections`)

  Provides text information without numerical analysis.

Each table and its contents is described in the following sections.

SNMP Trend Version Table (snmp_trend_version)

The `snmp_trend_version` table contains the version numbers. The data warehouse tools check this table to confirm the version has not changed. The `last_timestamp` is an Epoch time for the last time this function ran. This time is used as a starting point for the data warehouse tools.

Table 8-12 snmp_trend_version Table

Attribute Name	Data Type	Constraint	Description
`table_name`	VARCHAR	NOT NULL	The name of the data table that was previously exported.
`last_timestamp`	INTEGER		Time of the last exported detailed record.
`version`	INTEGER	NOT NULL	Software version number of the export tool that loaded the data.

SNMP Trend Description Table (snmp_trend_desc)

The `snmp_trend_desc` table provides a description of the monitored MIB variables and MIB expressions. Table entries describing MIB expressions are derived from the definitions in the `$OV_CONF/mibExpr.conf` file on UNIX and *`install_dir`*`\conf\mibExpr.conf` on Windows. Definitions for MIB expressions can be modified by editing `mibExpr.conf`. If you modify either the label or definition of an existing MIB expression in this file, a new `snmp_trend_desc` table entry is created the next time you execute `ovdwtrend`.

Table 8-13 snmp_trend_desc Table

Attribute Name	Data Type	Constraint	Description
`descript_id`	INTEGER	PRIMARY KEY	A unique identifier for this variable or expression.
`mib_oid`	VARCHAR	NOT NULL	A MIB variable object identifier in either dot notation with no spaces between digits and dots (with a leading dot) or a MIB expression label as defined in the `mibExpr.conf` file.
`units`	VARCHAR		The units of the data. For example, “units,” “units/sec,” “percent,” “errors/sec,” or “numJobs.”
`title`	VARCHAR	NOT NULL	A label for this MIB variable or MIB expression (for example, “`ifInOctets`”).
`descript`	VARCHAR		A description of this MIB variable or expression, as derived from the DESCRIPTION field of the MIB definition or the *`install_dir`*`\conf\mibExpr.conf` (for Windows) or `$OV_CONF/mibExpr.conf` (for UNIX) file. Descriptions greater than 512 characters are truncated.

Table 8-13 snmp_trend_desc Table (Continued)

Attribute Name	Data Type	Constraint	Description
`type`	VARCHAR	NOT NULL	Type of the data. Either an SNMP data type or "EXPRESSION" (if a MIB expression). Possible values are "COUNTER," "GAUGE," "INTEGER," "TIMETICKS," "EXPRESSION" or "STRING."
`mib_expr`	VARCHAR		If a MIB expression, the actual expression.

SNMP Trend Dataset Table (snmp_trend_dataset)

The entries in the `snmp_trend_dataset` table completely define a trend dataset by identifying the target host, MIB variable or expression, and instance. Note that the target host is keyed by its dataset ID, although the hostname is also provided. The hostname allows `ovdwtrend` to track topology object ID changes that occur when an object is deleted from and later added back to the topology database.

Table 8-14 snmp_trend_dataset Table

Attribute Name	Data Type	Constraint	Description
`topo_id`	VARCHAR	NOT NULL INDEX	The topology object ID of the host being monitored. This is a Unique Universal Identifier (UUID), and is the same ID as in the `nnm_nodes` topology table.
`descript_id`	INTEGER	NOT NULL INDEX	The `snmp_trend_desc` table ID for the MIB variable or expression being collected.
`instance`	VARCHAR	NOT NULL	The instance being monitored.
`dataset_id`	INTEGER	PRIMARY KEY	The unique identifier for this trend data set.

Table 8-14 snmp_trend_dataset Table (Continued)

Attribute Name	Data Type	Constraint	Description
`host_name`	VARCHAR	NOT NULL INDEX	Official hostname of the host being monitored.
`server_name`	VARCHAR		The node name of the NNM station that exported the data.
`t_last_raw_rec`	INTEGER		The timestamp of the last raw data record exported for the dataset.
`t_last_red_rec`	INTEGER		The timestamp of the last reduced data record exported for the dataset.
`title`	VARCHAR	NOT NULL	Name of the SNMP MIB expression that the data set represents (`ifInOctets`, for example).
`export_set`	VARCHAR		Name of the export set to which this dataset belongs. The value of this column is determined by the "`EXPORT_SET`" field in the SNMP data collector's meta-data files (those files in `$OV_DB/snmpCollect` on UNIX and *`install_dir`*`\databases\snmpCollect` on Windows ending in a "!"). NNM reporting datasets belong to the "`NNM_Reporting`" export set. Non-reporting datasets have a NULL `export_set` value by default.

SNMP Raw Trend Data Table (snmp_raw_trend)

The `snmp_raw_trend` table stores data collected from the SNMP environment. To allow for storage of rates (which is required for counter values and MIB expressions), the actual datapoint value is a float with at least 15 bits of precision. The `ovdwtrend` command writes to this table if you explicitly use the `-exportto raw` option.

NOTE Exporting raw trend data to the RDBMS consumes very large amounts of disk space and processing power. Therefore, HP OpenView recommends that instead of raw trend data, you export the reduced trend data (see "SNMP Reduced Trend Data Table (snmp_reduced_trend)" on page 170).

Table 8-15 snmp_raw_trend Data Table

Attribute Name	Data Type	Constraint	Description
`dataset_id`	INTEGER	NOT NULL INDEX	The identifier of this trend data set.
`sample_time`	VARCHAR	NOT NULL	The time at which this trend period ended (in a human readable format).
`sample_timestamp`	INTEGER	INDEX	The time in Epoch seconds which this trend period ended.
`period`	INTEGER	NOT NULL	The length of the period, in seconds.
`value`	FLOAT		The sampled data value.

SNMP Reduced Trend Data Table (snmp_reduced_trend)

The `snmp_reduced_trend` table offers an alternative to the high data space and performance demands of the SNMP Raw Trend Data Table, while still providing meaningful statistical information. By default, `ovdwtrend` writes to this table. By default, the `snmp_reduced_trend` table aggregates/summarizes data for one hour intervals.

The mean value is determined from the `data_sum` and `data_count` fields. The variance and standard deviation are calculated from the fields `data_sum`, `data_count`, and `sum_sqrs` by using the formula in Figure 8-4.

Figure 8-4 Formula for Calculating Standard Deviation

$$s = \{(1/(n-1))(\Sigma x^2 - n\bar{x}^2)\}^{1/2}$$

n is the data_count
$\bar{x}$ is the mean (data_sum/data_count)
Σx^2 is the sum_sqrs
s is the standard deviation

See the following directory for SQL scripts that include examples of how to calculate standard deviation for reduced data.

Windows:	`install_dir\contrib\NNM\trend`
UNIX:	`$OV_CONTRIB/NNM/trend`

Table 8-16 snmp_reduced_trend Data Table

Attribute Name	Data Type	Constraint	Description
dataset_id	INTEGER	NOT NULL INDEX	The identifier of this trend data set.
sample_time	VARCHAR	NOT NULL	The time at which this trend period ended (in a human readable format).
sample_ timestamp	INTEGER	NOT NULL INDEX	The time in Epoch seconds which this trend period ended. This timestamp is set to the exact end of the time period (2:59:59 for 2 p.m. to 2:59:59).
period	INTEGER	NOT NULL	The length of the reduction interval, in seconds.
min_value	FLOAT		The minimum of the data values in the reduced set.
max_value	FLOAT		The maximum of the data values in the reduced set.
data_count	INTEGER		The number of raw data values used to make up the reduced data.
data_sum	FLOAT		The summation of the data values in the reduced set.
sum_sqrs	FLOAT		The summation of the squares of the data values in the reduced set.

SNMP Aggregated Daily Trend Data Table (snmp_daily_trend)

The `snmp_daily_trend` table offers an alternative to the high data space and performance demands of the SNMP Raw Trend Data Table, while still providing meaningful statistical information. By default, `ovdwtrend` writes to this table. The mean value is determined from the `data_sum` and `data_count` fields. The variance and standard deviation are calculated from the fields `data_sum`, `data_count`, and `sum_sqrs` as shown in Figure 8-4.

See the following directory for SQL scripts that include examples of how to calculate standard deviation for reduced data.

Windows:	`*install_dir*\contrib\NNM\trend`
UNIX:	`$OV_CONTRIB/NNM/trend`

Table 8-17 snmp_daily_trend Data Table

Attribute Name	Data Type	Constraint	Description
`dataset_id`	INTEGER	NOT NULL INDEX	The identifier of this trend data set.
`descript_id`	INTEGER	NOT NULL	The `snmp_trend_desc` table ID for the MIB variable or expression being collected.
`host_name`	VARCHAR	NOT NULL	Official hostname of the host being monitored.
`mib_oid`	VARCHAR	NOT NULL	A MIB variable object identifier in either dot notation with no spaces between digits and dots (with a leading dot) or a MIB expression label as defined in the *`install_dir`*`\conf\mibExpr.conf` (for Windows) or `$OV_CONF/mibExpr.conf` (for UNIX) file.
`instance`	VARCHAR	NOT NULL	The instance being monitored.

Table 8-17 snmp_daily_trend Data Table (Continued)

Attribute Name	Data Type	Constraint	Description
sample_time	VARCHAR	NOT NULL	The time at which this trend period ended (in a human readable format).
sample_ timestamp	INTEGER	NOT NULL INDEX	The time in Epoch seconds which this time period ended. This timestamp is set to the exact end of the time period (23:59:59).
month	INTEGER		The month of the summarized data set.
year	INTEGER		The year of the summarized data set.
min_value	FLOAT		The minimum of the data values in the summarized data set.
max_value	FLOAT		The maximum of the data values in the summarized data set.
data_count	INTEGER		The number of data values in the summarized data set.
data_sum	FLOAT		The summation of data values in the summarized data set.
datapoint_ count	INTEGER		The total number of reduced datapoints. (Use data_count for the total number of raw datapoints before reducing the data.)
sum_sqrs	FLOAT		The summation of the squares of the data values in the summarized data set.

SNMP Aggregated Weekly Trend Data Table (snmp_weekly_trend)

The `snmp_weekly_trend` table offers an alternative to the high data space and performance demands of the SNMP Raw Trend Data Table, while still providing meaningful statistical information. By default, `ovdwtrend` writes to this table. The mean value is determined from the `data_sum` and `data_count` fields. The variance and standard deviation are calculated from the fields `data_sum`, `data_count`, and `sum_sqrs` as shown in Figure 8-4.

See the following directory for SQL scripts that include examples of how to calculate standard deviation for reduced data.

Windows:	*`install_dir`*`\contrib\NNM\trend`
UNIX:	`$OV_CONTRIB/NNM/trend`

Table 8-18 snmp_weekly_trend Data Table

Attribute Name	Data Type	Constraint	Description
`dataset_id`	INTEGER	NOT NULL INDEX	The identifier of this trend data set.
`descript_id`	INTEGER	NOT NULL	The `snmp_trend_desc` table ID for the MIB variable or expression being collected.
`host_name`	VARCHAR	NOT NULL	Official hostname of the host being monitored.
`mib_oid`	VARCHAR	NOT NULL	A MIB variable object identifier in either dot notation with no spaces between digits and dots (with a leading dot) or a MIB expression label as defined in the *`install_dir`*`\conf\mibExpr.conf` (for Windows) or `$OV_CONF/mibExpr.conf` (for UNIX) file.
`instance`	VARCHAR	NOT NULL	The instance being monitored.

Table 8-18 snmp_weekly_trend Data Table (Continued)

Attribute Name	Data Type	Constraint	Description
`sample_time`	VARCHAR	NOT NULL	The time at which this trend period ended (in a human readable format).
`sample_timestamp`	INTEGER	NOT NULL INDEX	The time in Epoch seconds which this time period ended. this timestamp is set to the exact end of the time period (Saturday 23:59:59).
`month`	INTEGER		The month of the summarized data set.
`year`	INTEGER		The year of the summarized data set.
`min_value`	FLOAT		The minimum of the data values in the summarized data set.
`max_value`	FLOAT		The maximum of the data values in the summarized data set.
`data_count`	INTEGER		The number of data values in the summarized data set.
`data_sum`	FLOAT		The summation of data values in the summarized data set.
`datapoint_count`	INTEGER		The total number of reduced datapoints. (Use `data_count` for the total number of raw datapoints before reducing the data.)
`sum_sqrs`	FLOAT		The summation of the squares of the data values in the summarized data set.

SNMP Aggregated Monthly Trend Data Table (snmp_monthly_trend)

The `snmp_monthly_trend` table offers an alternative to the high data space and performance demands of the SNMP Raw Trend Data Table, while still providing meaningful statistical information. By default, `ovdwtrend` writes to this table. The mean value is determined from the `data_sum` and `data_count` fields. The variance and standard deviation are calculated from the fields `data_sum`, `data_count`, and `sum_sqrs` as shown in Figure 8-4.

See following directory for SQL scripts that include examples of how to calculate standard deviation for reduced data.

Windows:	`*install_dir*\contrib\NNM\trend`
UNIX:	`$OV_CONTRIB/NNM/trend`

Table 8-19 snmp_monthly_trend Data Table

Attribute Name	Data Type	Constraint	Description
`dataset_id`	INTEGER	NOT NULL INDEX	The identifier of this trend data set.
`descript_id`	INTEGER	NOT NULL	The `snmp_trend_desc` table ID for the MIB variable or expression being collected.
`host_name`	VARCHAR	NOT NULL	Official hostname of the host being monitored.
`mib_oid`	VARCHAR	NOT NULL	A MIB variable object identifier in either dot notation with no spaces between digits and dots (with a leading dot) or a MIB expression label as defined in the *install_dir*`\conf\mibExpr.conf` (for Windows) or `$OV_CONF/mibExpr.conf` (for UNIX) file.
`instance`	VARCHAR	NOT NULL	The instance being monitored.

Table 8-19 snmp_monthly_trend Data Table (Continued)

Attribute Name	Data Type	Constraint	Description
`sample_time`	VARCHAR	NOT NULL	The time at which this trend period ended (in a human readable format).
`sample_timestamp`	INTEGER	NOT NULL INDEX	The time in Epoch seconds which this time period ended. this timestamp is set to the exact end of the time period (last day of last month 23:59:59).
`month`	INTEGER		The month of the summarized data set.
`year`	INTEGER		The year of the summarized data set.
`min_value`	FLOAT		The minimum of the data values in the summarized data set.
`max_value`	FLOAT		The maximum of the data values in the summarized data set.
`data_count`	INTEGER		The number of data values in the summarized data set.
`data_sum`	FLOAT		The summation of data values in the summarized data set.
`datapoint_count`	INTEGER		The total number of reduced datapoints. (Use `data_count` for the total number of raw datapoints before reducing the data.)
`sum_sqrs`	FLOAT		The summation of the squares of the data values in the summarized data set.

SNMP Text Collections Data Table

The `snmp_text_collections` table offers text information without numerical analyses.

Table 8-20 snmp_text_collections Data Table

Attribute Name	Data Type	Constraint	Description
`dataset_id`	INTEGER	NOT NULL	The identifier of this trend data set.
`sample_time`	VARCHAR	NOT NULL	The time at which this trend period ended (in a human readable format).
sample_ timestamp	INTEGER	NOT NULL	The time in Epoch seconds which this time period ended.
period	INTEGER	NOT NULL	The length of the reduction interval, in seconds.
value	VARCHAR		The sampled data value.

Event Data Schema

The events schema consists of 11 tables:

- Event Category table (`nnm_event_cat`)

  Pairs the category number of the event with a text description.
- Event Severity table (`nnm_event_sev`)

  Pairs the severity code of the event with a text description.
- Event Details table (`nnm_event_detail`)

  Provides a description of the events.
- Event Varbinds table (`nnm_event_varbinds`)

  Provides a list of the varbinds.
- Event Threshold table (`nnm_event_thresh`)

  Provides a list of the varbinds for a threshold event.
- Event Daily table (`nnm_event_daily`)

  Aggregates events for a 24 hour time period.
- Event Weekly table (`nnm_event_weekly`)

  Aggregates events for a weekly time period.
- Event Monthly table (`nnm_event_monthly`)

  Aggregates events for a monthly time period.
- Event Yearly table (`nnm_event_yearly`)

  Aggregates events for a yearly time period.
- Event Limits table (`nnm_event_limits`)

  Stores information about filters used to filter event types.

Each table and its contents is described in the following sections.

NNM Event Category Table (nnm_event_cat)

The `nnm_event_cat` table pairs the category number with a text description of the event type. Categories are defined by numbers 0 to 6. The categories are described in Table 8-21.

Table 8-21 nnm_event_cat Data Table

Attribute Name	Data Type	Constraint	Description
`category`	INTEGER	NOT NULL	The category number for the event.
`text`	VARCHAR	NOT NULL	The default categories for events include: 0 = Ignore 1 = Logonly 2 = Error Alarms 3 = Threshold Alarms 4 = Status Alarms 5 = Configuration Alarms 6 = Application Alert Alarms Categories 0 and 1 are always the same, but the other categories can be changed if you make changes to the *`install_dir`*`\conf\C\trapd.conf` (for Windows) or `$OV_CONF/C/trapd.conf` (for UNIX) file.

NNM Event Severity Table (nnm_event_sev)

The `nnm_event_sev` table pairs the severity code with a text description of the event severity. The severity options are described in Table 8-22.

Table 8-22 nnm_event_sev Data Table

Attribute Name	Data Type	Constraint	Description
`severity`	INTEGER	NOT NULL	The severity number for the event.
`text`	VARCHAR	NOT NULL	The severities for events include: normal, warning, minor, major, critical, and unknown.

NNM Event Details Table (nnm_event_detail)

The `nnm_event_detail` table contains the raw metadata for the event.

Table 8-23 nnm_event_detail Data Table

Attribute Name	Data Type	Constraint	Description
`event_uuid`	VARCHAR	PRIMARY KEY	The identifier for the event.
`event_timestamp`	INTEGER	NOT NULL INDEX	The time in Epoch seconds when this event occurred.
`category`	INTEGER	NOT NULL INDEX	The category of the event as described by the `nnm_event_cat` table.
`nodename`	VARCHAR	NOT NULL INDEX	The name of the node that collected this event.
`application_id`	INTEGER		The software source of the event that indicates which NNM application or process generated the event.
`message`	VARCHAR		The formatted string describing the event as described in *`install_dir`*`\conf\C\trapd.conf` (for Windows) or `$OV_CONF/C/trapd.conf` (for UNIX) file.
`severity`	INTEGER	NOT NULL INDEX	The numeric severity of the event as described by the `nnm_event_sev` table.
`event_oid`	VARCHAR	NOT NULL INDEX	The object identifier of the event.
`ov_objectid`	VARCHAR		The HP OpenView object ID, or 0, if not available.
`protocol`	VARCHAR		The SNMP protocol type.

Table 8-23 nnm_event_detail Data Table (Continued)

Attribute Name	Data Type	Constraint	Description
event_type	VARCHAR	NOT NULL	The enterprise-specific event object identifier.
ip_address	VARCHAR		The IP address for the NNM management station reporting the event.
trapsource	INTEGER		The nodename for the node sending the trap.
trap_name	VARCHAR		Name of the SNMP trap that caused the event (for example, OV_Node_Down, OV_Node_Up).
pid	INTEGER		Process ID for the process sending the event.
forward_ address	VARCHAR		IP address of another NNM pmd to forward event to.
event_source	VARCHAR		The hostname for the source of the event.
event_time	VARCHAR		The time the event was received in a readable format.
number_ varbinds	INTEGER		The number of records associated with the event in the nnm_event_varbinds table.

NNM Event Varbinds Table (nnm_event_varbinds)

The `nnm_event_varbinds` table contains the additional data associated with the event, as defined in the `trapd.conf` file for the specified event OID.

Table 8-24 nnm_event_varbinds Data table

Attribute Name	Data Type	Constraint	Description
`event_uuid`	VARCHAR	NOT NULL INDEX	The identifier for the event.
`varbind_ sequence`	INTEGER	NOT NULL	Sequence of varbinds of the event as defined in *`install_dir`*`\conf\C\trapd.conf` (for Windows) or `$OV_CONF/C/trapd.conf` (for UNIX) file. Use an offset of +1 to match the `trapd.conf` values
`event_oid`	VARCHAR	NOT NULL INDEX	The varbind object identifier.
`val_len`	INTEGER		The length of the string value.
`type`	VARCHAR		Specifies the field (below) in which to find the data based on the data type.
`type_string`	VARCHAR		The varbind string value.
`type_integer`	INTEGER		The varbind integer value.
`objid`	VARCHAR		The varbind OID value.
`unsigned32`	INTEGER		The varbind 32-bit integer value.
`unsigned64`	INTEGER		The varbind 64-bit integer value.

NNM Event Threshold Table (nnm_event_thresh)

The `nnm_event_thresh` table contains the varbinds for the threshold events.

Table 8-25 **nnm_event_thresh Data Table**

Attribute Name	Data Type	Constraint	Description
`event_uuid`	VARCHAR	PRIMARY KEY NOT NULL	The identifier for the event.
`event_oid`	VARCHAR	NOT NULL INDEX	The object identifier of the event.
`event_timestamp`	INTEGER	NOT NULL INDEX	The time in Epoch seconds when this event occurred.
`event_time`	VARCHAR		The time the event was received in a readable format.
`application_id`	INTEGER		The software source of the event that indicates which NNM application or process generated the event.
`nodename`	VARCHAR	NOT NULL INDEX	The name of the node that collected this event.
`ov_object_id`	VARCHAR		The HP OpenView object ID, or 0, if not available.
`mib_oid`	VARCHAR	NOT NULL	A MIB variable object identifier in either dot notation with no spaces between digits and dots (with a leading dot) or a MIB expression label as defined in the *`install_dir`*`\conf\mibExpr.conf` (for Windows) or `$OV_CONF/mibExpr.conf` (for UNIX) file.

Table 8-25 **nnm_event_thresh Data Table (Continued)**

Attribute Name	Data Type	Constraint	Description
`collection_ name`	VARCHAR	NOT NULL	The name (label) of the data collector collection whose threshold violation triggered the threshold event.
`instance`	VARCHAR	NOT NULL	The instance being monitored.
`severity`	INTEGER	NOT NULL	The numeric severity of the event as described by the `nnm_event_severity` table.
`threshold`	FLOAT	NOT NULL	The numeric value that you set as the standard for comparing collection samples to identify threshold violations.
`sample`	FLOAT	NOT NULL	Numeric value collected for the MIB or MIB expression.
`high_sample`	FLOAT	NOT NULL	Highest sample (peak) value collected for node and instance.
`high_time`	VARCHAR	NOT NULL	The time that `high_sample` was collected.
`low_sample`	FLOAT	NOT NULL	Lowest sample value collected for the node and instance.
`low_time`	VARCHAR	NOT NULL	The time that `low_sample` was collected.
`threshold_op`	VARCHAR	NOT NULL	Operator (>, <, ==, >=, <=, !=) used to compare the sample with the threshold.
`threshold_ count`	INTEGER		The number of consecutive times the reset value was exceeded before threshold rearm.

NNM Event Aggregation Tables

The `nnm_event_daily`, `nnm_event_weekly`, `nnm_event_montly`, and `nnm_event_yearly` tables contain aggregated data for a specified period of time. These four event aggregation tables contain the same fields as described in Table 8-26.

Table 8-26 nnm_event_(daily, weekly, monthly, or yearly) Data Table

Attribute Name	Data Type	Constraint	Description
`nodename`	VARCHAR	PRIMARY KEY	The name of the node that collected this event. The primary key.
`event_oid`	VARCHAR	NOT NULL INDEX	The object identifier of the event.
`event_cnt`	INTEGER	NOT NULL	Number of times the associated event occurred for the nodename or event_oid within time period (of 1 day).
`event_timestamp`	INTEGER	INDEX	The time in Epoch seconds when this event occurred.
`event_time`	VARCHAR		The time the event was received in a human readable format. This timestamp is set to the exact end of the time period (day, month, week, or year as appropriate 23:59:59).
`day`	INTEGER		The day for this aggregated event data. The value is set to zero in all tables except daily.
`month`	INTEGER		The month for this aggregated event data.
`year`	INTEGER		The year for this aggregated event data.

NNM Event Limits Table (nnm_event_limits)

The nnm_event_limits table contains data about filters to be used to filter event types when reducing the event data. Use ovdweventflt to add filters to this database.

Table 8-27 nnm_event_limits Data Table

Attribute Name	Data Type	Constraint	Description
nodename (synonym: ip_hostname)	VARCHAR	NOT NULL	The name of the node to be filtered. The primary key.
event_oid	VARCHAR	NOT NULL INDEX	The object identifier of the event.

always_never	VARCHAR		Filtering indicator for whether to always or never filter.
			Y = always filter; do not export
			N = never filter; export

limit	INTEGER		The maximum number of rows for this node or MIB OID to be exported on a daily basis. This field is used only when the always_never field is NULL.

Common Data Schema

This schema holds operational data, such as timestamps for the control of data export operations. During the previous event export operation, the timestamp for the last event to be exported is saved in this table. The next event export operation continues the export after this saved timestamp.

Table 8-28 nnm_event_data Data Table

Attribute Name	Data Type	Constraint	Description
`table_name`	VARCHAR		The name of the data table that was previously exported.
`last_timestamp`	INTEGER		The time of the last exported event.
`version`	INTEGER		Software version number of the export tool that loaded the data.

9 Backup and Recovery

Backing up the data warehouse is important. This chapter explains what parts of the data warehouse you must back up regularly in order to recover data successfully.

Backup and Recovery

Backing up your database and its environment on a regular basis is important. You should back up the following files regularly:

- Data files of the database itself.
- Control files which identify a database and log files associated with the database.

Backing up the database and its environment allows you to:

- Restore the database and its environment as often as necessary.
- Copy databases from one system to another.
- Limit disk space used by log files.

If you are using an external database, check your vendor's database documentation for specific instructions for completing a database recovery.

The backup instructions in this manual consider only the integrity requirements for NNM. Other HP OpenView products, as well as any third-party applications that use the data warehouse databases, may provide their own methods for database backups.

As you design a backup strategy to meet your needs, use the NNM-based procedures described here as a guideline only. Be sure to incorporate personal requirements and those of third-party applications whose backup methods you want to use.

You can use the following methods to backup data in the data warehouse:

- Default `solid.ini` scheduled backup (`At=23:00 backup`).
- NNM backup using `ovbackup.ovpl`.
- Backup of the Oracle database.
- Backup of the Microsoft SQL Server database.
- Independent backup using the data warehouse commands to back up any of the HP OpenView-supported databases.

The embedded database creates log files. The log files have the format `sol#####.log` and are located in the following directory:

Windows:	`*install_dir*\databases\analysis\default\log`
UNIX:	`$OV_DB/analysis/default/log`

These log files can often be useful when attempting to recover a database environment. The procedures for using log files to assist in recovering a database environment depend on several factors:

- Was the connection to the database interrupted or was data lost or damaged?
- What type of recovery is needed?
- How many log and control files are needed to execute the recovery?

Default solid.ini Scheduled Backup

By default, the data warehouse performs an internal online checkpoint and backup on a daily basis at a predetermined time. See the following file for specific information.

Windows:	`*install_dir*\databases\analysis\default\solid.ini`
UNIX:	`$OV_DB/analysis/default/solid.ini`

Before starting the `solid.ini` scheduled backup, a checkpoint is created automatically. This checkpoint guarantees that the state of the database backup is from the instant the online backup process was started.

The `solid.ini` scheduled backup copies the following files to the backup directory:

- Database files
- Configuration file (`solid.ini`)
- Log files modified or created after the previous backup
- License file (`solid.lic`)

After a successful online backup, the unnecessary log files are deleted from the original log directory, and the backup copy is placed in the following directory:

Windows:	`*install_dir*\databases\analysis\default\backup`
UNIX:	`$OV_DB/analysis/default/backup`

This backup copy remains unchanged until the next online backup occurs.

Correcting a Failed Backup

If the online backup fails, an error message appears in the following file:

Windows:	`*install_dir*\databases\analysis\default\solerror.out`
UNIX:	`$OV_DB/analysis/default/solerror.out`

Correct the cause of the error and try the backup again. The most common causes for failed online backups are:

- The backup media is out of disk space.
- The backup directory does not exist.
- A database directory is defined as the backup directory.

If you do not perform online backups on a regular basis, the files in the log directory grow infinitely, and database performance is severely impacted.

NNM Backup Using ovbackup.ovpl

To use `ovbackup.ovpl` to backup the data warehouse, run:

`ovbackup.ovpl -analytical [-d` ***`destination directory`*****`]`**

Refer to the *ovbackup.ovpl*(1) manpage or reference page for more information.

NOTE If you choose to run `ovbackup.ovpl`, you must deactivate the default `solid.ini` scheduled backup. Do not do both or neither. Doing both results in backups that may not restore properly. Doing neither may cause severe performance problems with the embedded database.

Use the following procedure to deactivate the default `solid.ini` scheduled backup. You can find the files in this procedure in the following directory:

Windows:	*`install_dir`*`\databases\analysis\default`
UNIX:	`$OV_DB/analysis/default/`

1. Copy the `solid.ini` file to `solid.ini.old` file.
2. Edit the `solid.ini` file.
3. Comment out the `At=`*`time`*` backup` entry, by inserting a "`;`" at the beginning of the line. For example:

   ```
   ;At=23:00 backup
   ```
4. Save the `solid.ini` file.

The embedded database is backed up only when you run the `ovbackup.ovpl` command. If you stop use of `ovbackup.ovpl` at a future time, you should reinstate the default backup by copying the `solid.ini.old` file back to `solid.ini`.

The `ovbackup.ovpl` command instructs the `solid.ini` scheduled backup to begin.

Restoring Backups

There are two ways to restore a backup. You can either:

- Return to the state when backup (consolidation) was created.
- Recover a backup database to the current state. Use log files to add data that was inserted or updated after the consolidation was made.

To return to the state when the backup was made:

1. Run **`ovstop ovdbcheck`** to shut down the embedded database.
2. Delete all log files from the following directory:

Windows:	`*install_dir*\databases\analysis\default\log`
UNIX:	`$OV_DB/analysis/default/log`

   Examples of default log file names are `sol00001.log`, `sol00002.log`.
3. Copy the database files from the `/default/backup` directory to the database file `/default` directory.
4. Run **`ovstart ovdbcheck`** to start the embedded database.

This method will not perform any recovery because no log files exist.

To recover the database to the current state:

1. Run **`ovstop ovdbcheck`** to shut down the embedded database.
2. Copy the database files from the `/default/backup` directory to the database file `default` directory.
3. Copy the log files from the `/default/backup` directory to the `/default/log` file directory. If there are log files with the same file names, do *not* replace those log files in the log file directory with log files from the backup directory.
4. Run **`ovstart ovdbcheck`** to start the embedded database.

The embedded database will automatically use the log files to perform a roll-forward recovery.

If you are using the embedded database, the automated backup of the database operates during the analytical phase. The default `solid.ini` scheduled backup facility occurs on a daily basis.

Backup of an Oracle Relational Database

Your `$ORACLE_HOME` directory contains some files that need backing up on a regular basis. These files are critical to a successful recovery (for example, control files). HP OpenView recommends that you back up control files on a regular basis.

Refer to the Oracle database documentation for further information on backup and recovery strategies. Another useful reference is the *Oracle Backup and Recovery Handbook*, by Rama Velpuri (Oracle Press:1995).

See your Oracle reference manuals for details.

Backup of a Microsoft SQL Server Database

You should complete backups after you create the Microsoft SQL Server database and on a regular backup schedule thereafter. To backup Microsoft SQL Server, either:

- Use `SQL Enterprise Manager` to define and schedule the backups.
- Backup the database manually at any time.

Refer to the Microsoft SQL Server documentation to obtain more information about defining and performing backups.

Independent Backup

You can use the data warehouse commands, `ovdwloader` and `ovdwunloader`, to make an independent backup copy of the data in the data warehouse. This backup method is independent of the database storing the data. The commands to copy event and trend data are:

```
ovdwunloader -event
```

```
ovdwunloader -trend
```

These commands make a master copy of the entire database that contains a header record followed by ASCII records. The format of the new file is:

- A header record contains the table name, number of records following, and the schema version number.
- One record is written for each logical row in the data warehouse.
- ASCII columns are separated by white space.
- Data for multiple tables can be concatenated into one input stream.

You can use `ovdwloader` to load the copied data into one of the database products. `ovdwloader` checks that the number of records is the same as the number of records specified in the header record. If there is a discrepancy in the number of records, the data is not loaded.

A NNM Data Warehouse Troubleshooting

This section describes problems you may encounter while using the data warehouse. If you are using an external database product and the suggested solutions in this chapter do not help to solve the problem, refer to the vendor's database documentation for additional help.

Identifying the Problem Using ovdbdebug (UNIX Only)

The data warehouse provides the `ovdbdebug` validation tool that you can use to determine whether the database is functional. This tool supports all databases and ODBC. The `ovdbdebug` command:

- Displays database configuration information including:
  - Contents of the `$ODBCINI`, `$OV_CONF/ovdb.conf`, and `ovenvars.conf` files (for all databases).
  - Contents of `$ORACLE_HOME`, `$ORACLE_BASE`, `$ORACLE_SID`, the Oracle release, and the existence of parameter file (for the Oracle database only).
- Attempts to connect to the respective database and indicates whether the database server is up and can be connected to the data warehouse. If the connection fails, `ovdbdebug` returns an error message from the database product.

NOTE To execute the `ovdebug` command successfully, `/usr/ucb` must be in the `root` path.

If you are using an external database product and `ovdbdebug` cannot connect to the database, you may have a problem with the external database, the ODBC driver configuration, or the database product's network software. Contact your database vendor for assistance.

If you are unable to solve your problems with the data warehouse, contact Hewlett-Packard support. Be sure to have the following information available:

- Contents of `$OV_CONF/ovdbconf` (if this file exists; Oracle only).
- Output generated by `ovdbdebug`.

The text below is an example of an `$OV_PRIV_CONF/ovdbconf` file for a UNIX system using the Oracle database:

```
DB_VENDOR Oracle
DB_NAME openview
DB_RELEASE 8.0.4
DB_TIME_STAMP "Mon Oct 19 08:43:25 MDT 1998"
DB_USER ovdb
ORACLE_SID openview
ORACLE_HOME /opt/u01/home/oracle/product/8.0.4_11.0
ORACLE_BASE /opt/u01/home/oracle
DBA_USER oracle
DATA_DIR /u01/oradata/openview
CREATE_DIR /opt/u01/home/oracle/admin/openview/create
INDEX_DIR /u01/oradata/openview
ADMIN_DIR /opt/u01/home/oracle
OS_AUTHENT_PREFIX
CHARACTER_SET WE8ISO8859P1
BASE_DATA_TS_SIZE 25
BASE_INDEX_TS_SIZE 5
DATA_TS_SIZE 25
INDEX_TS_SIZE
TEMP_TS_SIZE 2
DATA_TS_EXTENT_SIZE 2
DATA_TS_MAX_SIZE 500
INDEX_TS_EXTENT_SIZE
ECHO_CMD echo
PROMPT TRUE
DBA_PROGRAM svrmgrl
OV_USER ovdb
DBA_LOGFILE /var/opt/OV/share/log/svrmgrl_log
ORACLE_BASE_REV 8
ORACLE_SECOND_REV 0
NLS_LANG american_america.WE8ISO8859P1
```

Using ovdbcheck

The `ovdbcheck` command validates whether the database is working properly.

Run the command:

```
ovdbcheck -info
```

`-info` returns information about the ODBC driver type, ODBC driver version, database type, and database version.

If the database cannot be connected, errors are sent to stderr or to the command prompt.

Examining log Files for Problems

Log files provide an historical perspective of the use and problems in the data warehouse. You can find the log files in the following directory:

Windows:	`*install_dir*\log`
UNIX:	`$OV_LOG`

This directory contains the following log files:

- `embeddedDbConf.log` and `externalDbConf.log`

  Includes the information for configuring the data warehouse and instantiating the schema.

- `ovdbcheck.log`

  Includes the `ovdbcheck` failures.

- `ovdwtrend.log`

  Includes export, summarization, and trim tools operations.

- `ovdwtopo.log`

  Includes topology export operations.

- `ovdwevent.log`

  Includes event, export, and trim operations.

Read the log files to assess whether the data warehouse is functioning properly.

Troubleshooting Strategies for Trend Data Storage/Access Problems

How to Recover from ovdwtrend Errors

When `ovdwtrend` encounters an error, it restores the source database (`*install_dir*\databases\snmpCollect` (for Windows) `$OV_DB/snmpCollect` (for UNIX) by default) to a consistent state. If you used the `-delpriorto` option to delete source data, only data successfully exported is deleted. The remaining data is restored to the source database. After addressing the cause of the error, you may resume export by re-executing `ovdwtrend`. `ovdwtrend` ensures that data exported in the original execution is not exported again.

When the trend data utilities encounter a database error, an explanation of the task in progress, as well as the database error message and error code, is printed to both `stderr` and `*install_dir*\log\ovdwtrend.log` (for Windows) and `$OV_LOG/ovdwtrend.log` (for UNIX). When this occurs, refer to the database documentation to rectify the problem. Then, re-execute the original trend data command to complete the operation.

Avoiding Duplicate and Overlapping Data Records

In the SNMP trend database, the presence of records with overlapping collection periods makes the data more difficult to analyze. These duplicate records also waste disk space. For a specific dataset (node, MIB variable, and instance combination), `ovdwtrend` only exports “new” data to prevent the storage of duplicate or overlapping data. “New” data has a more recent sample time than existing data in the database. If “old” data is encountered, `ovdwtrend` prints the following message after exporting data:

```
NN records skipped because they are older than the most recent record (for that
dataset) in the database.
```

This message is often the result of repeated executions of `ovdwtrend` without using the `-delpriorto` (delete source) option. This message indicates that `ovdwtrend` is not exporting data that was previously exported. If you are exporting data collected by multiple NNM stations (for example, distributed collection stations), this message may indicate that two or more stations are collecting on the same data.

Troubleshooting Database Connection Problems

If you have problems connecting to the embedded database, follow the troubleshooting suggestions below:

1. Run **`ovstatus ovdbcheck`** to determine whether the embedded database manager, `ovdbcheck`, is running. `ovdbcheck` monitors the database server. If the database server is not functioning correctly, `ovdbcheck` terminates.
2. Run the Task Manager (Windows operating systems) or **`ps`** (UNIX) to determine whether the embedded database server process, `ovdbrun`, is running.
3. Run **`ovdbcheck`** from the command line. If it fails, try to connect directly to the database with ***`install_dir`*`\bin\solsql "tcpip 2690"`** (Windows operating systems) or **`$OV_BIN/solsql "tcpip 2690"`** (UNIX). If this connection is successful, but `ovdbcheck` fails, there may be an ODBC driver problem.
4. Check the ODBC driver:

   For Windows:

   Run the ODBC Administrator to determine if the ODBC data source in use is listed in the `System DSN` tab.

   For UNIX:

   - Validate that the `$OV_CONF/analysis/system_odbc.ini` file exists.
   - Validate that the `"tcpip 2690"` and the driver file indicated by the `Driver=` exist and have a mode of 555.
5. Verify that the parameters provided in *`install_dir`*`\conf\analysis\ovdw.conf` (for Windows) or `$OV_CONF/analysis/ovdw.conf` (for UNIX) are correct:

   - `user:` *`username`* is the default database user that the data warehouse tools use.

- `password:` *`passwd`* is the default password that the data warehouse tools use for the user. The password is encrypted. To validate the correct password, you must run the data warehouse tools.
- `dbname:` *`ODBC_datasource`* is the default ODBC data source. For the embedded database, the ODBC data source is the name of the "server listener," and it should be listed in the *`install_dir`*`\databases\analysis\default\solmsg.out` (for Windows) or `$OV_DB/analysis/default/solmsg.out` (for UNIX) file to be started every time the embedded database is started.
- `db: [yes|no]` is the embedded database startup indicator.

  If `yes` is specified, ovdbcheck starts the embedded database.

  If no is specified, `ovdbcheck` only validates that it can connect to the ODBC data source, as indicated by `dbname:`.

These parameters are maintained by `ovdwconfig.ovpl`. Run `ovdwconfig.ovpl` to make changes to these parameters.

6. Check that the environment variable information required by ODBC and the configured database is correct:

   - On UNIX, the `$OV_CONF/analysis/ovdwenvs.conf` file should contain the ODBC parameter file:

     ```
     ODBCINI=/etc/opt/OV/share/conf/analysis/
     system_odbc.ini
     ```

   - On UNIX, the following paths should be included to allow the ODBC drivers to be loaded when the ODBC applications are started:

     HP-UX: `SHLIB_PATH=/opt/OV/iodbc/lib`

     Solaris and Linux: `LD_LIBRARY_PATH=/opt/OV/iodbc/lib`

   - If you are using Oracle, an additional path, `$ORACLE_HOME/lib`, is appended to the environment variable described above so that `libclntsh.[so|sl]` can be loaded by the Oracle ODBC driver.
   - Specify `ORACLE_HOME` to allow the data warehouse tools to be run without each user having to explicitly specify this environment variable. Optionally, if you are using Oracle:

     `ORACLE_HOME=`*`value of $Oracle_HOME`*

The environment variables are maintained by `ovdwconfig.ovpl` and are indicated by the `env Variable=Value` parameter.

7. Check that the following ODBC data sources were created during the installation of NNM:

   - On HP-UX and Solaris:

     For the embedded database: `"tcpip 2690"`

     For Oracle, the NNM-supplied Oracle data source: `OVoraWire`

   - On Linux:

     For the embedded database: `"tcpip 2690"`

     For Oracle 8.1.7 and 9.2.0, the NNM-supplied Oracle data source: `OVoraWire`

   - On Windows:

     For the embedded database: `ovdbrun`

     For Oracle and Microsoft SQL Server, the ODBC data source are configured by the `ODBC Data Source Administrator` as described in Chapter 3, “Installation and Configuration.” The ODBC drivers for these databases are not supplied by HP.

Troubleshooting for the Embedded Database

Problems Starting Database with ovstart ovdbcheck

If you have problems starting the embedded database with `ovstart ovdbcheck`, follow the troubleshooting suggestions below:

1. Check the following file for specific error messages.

Windows:	`*install_dir*\log\ovdbcheck.log`
UNIX:	`$OV_LOG/ovdbcheck.log`

2. Check that the following files exist in the *`install_dir`*`\databases\analysis\default` (Windows) or `$OV_DB/analysis/default` (UNIX) directory:

   - `solid.db` is the actual database.
   - `solid.ini` is the embedded database parameter file.
   - `solid.lic` is the embedded database license file.

3. Consult the following files in *`install_dir`*`\databases\analysis\default` (Windows) or `$OV_DB/analysis/default` for information that may give an indication of the problem:

   - `solmsg.out` records normal activities, such as database startup/shutdown, user connections, and times of checkpoints and backups.
   - `solerror.out` records embedded database server errors.

Repairing a Corrupt Database

If you believe that your database is corrupt, the following procedures can help recover your database.

Check the database consistency:

1. Make sure the Solid server is not running by typing **`ovstop ovdbrun`**.
2. Run **`$OV_BIN/ovdbrun -x testindex`** to test the database integrity.

3. If you get an Internal Error message, it is likely that your database is corrupt. If not, it is likely that your license file or log files are corrupt or out of sequence.

If you cannot get your embedded database to start, try recovering the database by restoring the `solid.db` to the current state after the failure from your last backup:

1. Copy `solid.db` from the backup directory to the database directory:

   copy `$OV_DB/analysis/default/backup/solid.db` to `$OV_DB/analysis/default/solid.db`

2. Copy the log files from the backup directory to the log file directory:

   copy `$OV_DB/analysis/default/backup/sol*.log` to `$OV_DB/analysis/default/log/`

NOTE Do *not* overwrite existing log files by the same name.

3. Start the Solid embedded engine by typing **`ovstart ovdbcheck`**. Roll-Forward recovery is done automatically.

If the above procedure fails and you still cannot get your embedded database to start, try returning to your backup state:

1. Delete log files from the log directory.
2. Copy `solid.db` from the backup directory to the database directory:

   copy `$OV_DB/analysis/default/backup/solid.db` to `$OV_DB/analysis/default/solid.db`

3. Restart the embedded database with `ovstart ovdbcheck`.

If the procedures described above do not correct the problem, you have two options: recreate the database, or contact your HP support person for help in recovering your data. Recreating the database will reinitialize the embedded database. Your existing data will no longer be available in the new database. Data will still exist in the old database for future attempts to recover.

To recreate the embedded database (you must be `root` on UNIX or `Administrator` on Windows):

1. Backup the existing database and all associated files under `$OV_DB/analysis/default/` to a safe place as a precaution.
2. Change directory to `$OV_DB/analysis/default/`
3. Delete `solid*.db`, `./sol*.log`, and `./log/*`, `./backup/*` files.
4. Run **`$OV_BIN/ovdbrun -x exit`** while in the `$OV_DB/analysis/default` directory. You will be prompted for the default database user and passwords (`ovdb`, `ovdb`). You will also be prompted for the default system catalog name (`ovdb`).

   Performing this step creates an empty database (`solid.db`).
5. Populate the database with the NNM schema:

   ```
   $OV_BIN/ovdwconfig.ovpl -type embedded -load
   ```
6. Start the server by typing **`ovstart ovdbcheck`**.

Troubleshooting for the Oracle Database

Connection Problem

If any of the data warehouse utilities cannot connect to the database, follow the instructions below:

1. Enter:

   `sqlplus`

2. Respond to the user and password prompts.

   If the connection:

   - Does succeed and one of the data warehouse utilities reported the error:

     Run **`ovdbcheck -info`** to verify that the ODBC environment is correctly configured.

   - Does not succeed and `sqlplus` fails:

     a. Verify that the Oracle Server and SQL*Net Listener are running.

     b. Verify the user and password are correct.

     c. Verify the user `ovdb` has the `CREATE SESSION` privilege.

     d. Verify that your `ORACLE_HOME` and `ORACLE_SID` are set correctly in the environment.

3. For UNIX only, as `root`, run the command:

   `ovdbdebug -o`

Configuration Problem

If the `ovdwconfig.ovpl` utility fails to create the database tables correctly:

1. Look up the reported Oracle error in the Oracle messages documentation or use the Oracle `oerr` utility by entering the following command:

`oerr` ***`messagetype messagenumber`***

`messagetype` is the type of Oracle error message.

`messagenumber` is the number of the Oracle error message.

2. Verify the following items.

   - The specified database user has the following privileges:

     ```
     CREATE SESSION
     ALTER SESSION
     CREATE TABLE
     CREATE PUBLIC SYNONYM
     DROP PUBLIC SYNONYM
     ```

   - The specified database user has sufficient sizes for all valid default, temp, `OV_DATA`, and `OV_INDEX` tablespaces.
   - The specified database user has sufficient (or unlimited) quota on the default, temp, `OV_DATA`, and `OV_INDEX` tablespaces.

NOTE Monitor your database capacity frequently to avoid filling your database or transaction log.

- Allow at least 20% free space in your database at all times.
- Schedule the use of the `trim` options that are available with `ovdwtrend` and `ovdwevent`.

Refer to Chapter 6, "Export Utilities," for more information about these utilities.

Troubleshooting for the Microsoft SQL Server

Connection Problem

If one of the data warehouse utilities cannot connect to the Microsoft SQL Server database, follow the suggestions below.

This problem may be caused by terminating one of the data warehouse export utilities prematurely and attempting to restart the export process before Microsoft SQL Server has completed the database recovery process.

Make sure that Microsoft SQL Server service is running. Start SQL Service Manager, select SQL Server from the pull-down menu and click on the start icon.

Data Insert Problem

If the data warehouse export utilities fail to insert new data in the data warehouse, follow the suggestions below:

This problem may occur when the data warehouse `openview` database becomes full.

1. Verify the cause of the problem by viewing the last few records in the export utility log file:

   For problems with `ovdwtopo`, see `*install_dir*\log\ovdwtopo.log`.

   For problems with `ovdwevent`, see `*install_dir*\log\ovdwevent.log`.

   For problems with `ovdwtrend`, see `*install_dir*\log\ovdwtrend.log`.

2. View the SQL Server transaction log file, `\MSSQL\LOG\ERRORLOG`, to obtain detailed information about the last SQL activity that took place on your database.

If your database is full, follow your Microsoft SQL Server procedure for expanding your database.

NOTE

Monitor your database capacity frequently to avoid filling your database or transaction log.

- Allow at least 20% free space in your database at all times.
- Schedule the use of the `trim` options that are available with `ovdwtrend` and `ovdwevent`.

Refer to Chapter 6, “Export Utilities,” for more information about these utilities.

Troubleshooting the Web Reporting Interface

Missing Reports

- System configuration issues that cause this problem may be addressed by the system administrator. See the recommended configuration section of this manual.
  - Disk filling up prevents the storage of information and reports.
  - Not enough swap configured on the system prevents reporting processes from running.
  - E-mail not being delivered (with generated reports) may result from the destination host for e-mail or the SMTP Host not being configured with an SMTP server.
  - Kernel configuration items:
    - If you're using Oracle on a UNIX system, you may need to increase the `maxusers` parameter.
    - If you're running on an HP 9000 Series 800, you may need to increase your cache size. Refer to "Improving Performance" on page 227.
    - If you're using Oracle 7.3.4 on Solaris 2.7, increase `SHMMAX` to 4294967295.
- Has the NNM management station been shut down during the nighttime hours? This would prevent the SNMP data collector (and other processes) from gathering data on nodes for reports. It also prevents data warehouse export and report generation tasks from running at their prescribed times. Run `ovstatus` to verify the current operation of the NNM processes.
- Verify the reports you expect to see are scheduled in the Report Configurator by selecting

  `Options->Report Configuration`

  Double click `View Scheduled Reports`.
- Reports are generated beginning at midnight for the previous day. Has it been 24 hours since you configured the report?

- NNM daily performance reports require that `snmp_reduced_trend` data reside in the data warehouse for at least a full day. If you have configured NNM performance reports and use the `ovdwtrend -trim` command, be sure to specify a `-trimpriorto` value greater than 24 hours.

Performance reports show up in the Report Presenter but contain a message that there is no data on which to report

1. Interface error reports show data only for interfaces with error rates greater than or equal to 0.01%. Perhaps you have no interfaces that meet that criteria.
2. Verify that NNM is correctly configured for SNMP connection to the devices on which you're reporting by running

   `snmpwalk` ***`<node name>`*** **`system`**

   If this command returns system level information, SNMP is properly configured. If it returns an error message, or times out, the node (or its SNMP agent) is down or NNM is not configured with the SNMP get community string of the target node. To configure the community string, select

   `Options->SNMP Configuration`

   from the `ovw` menu.
3. Verify that IP address wildcards or SNMP SysObjectIDs used in configuring the performance reports are correct.

Threshold Violation reports continuously come up empty

1. Verify that thresholds are configured for SNMP. To verify the current settings select

   `Options->Data Collections & Thesholds:SNMP`

2. NNM adds data collection on MIB expressions beginning with NNM_ and with an origin of `Reporting`. Verify that you have manually set threshold values for these expressions and that the status is `Collecting`.

3. Verify that attainable thresholds and rearm values are set to activate the events that create these reports. If you have been collecting on NNM_* MIBs, a convenient way to ascertain a reasonable threshold is to examine the current values by running

snmpColDump ***<collection-to-set-threshold-upon>***

For example,

```
snmpColDump NNM_IfInOctets_defGroup.1
```

Reviewing Report Schedules Using request_list

You can use the `request_list` command to see when Reporting will run the export tasks (events, topology, and trend). You must run this command as user `root` or `bin` on UNIX or with Administrator capabilities on Windows.

Without arguments, this command lists the tasks that the reporting scheduler (`ovrequestd`) has scheduled, and their status. Passing an attribute name causes `request_list` to also list the value of that attribute for each task. The `schedule` attribute produces the schedule for the task. Look for the tasks with the word “export” in them. For example, on UNIX, the following command tells you the schedule for just the export tasks:

```
request_list schedule | grep export
```

The schedule returned is in the same format as the UNIX `cron` utility. Each schedule is a string of five fields, separated by spaces or tabs. The fields represent the minutes (0-59), hours (0-23), monthdays (1-31), months (1-12), and weekdays (0-6, 0=Sunday) that the task will be run, in that order. An asterisk represents all legal values. For example, the `request_list` command above might return the following output:

```
C/export/events  ACTIVE 30 * * * *

C/export/topology  ACTIVE 0 0,12 * * *

C/export/trend  ACTIVE 40 0,12 * * *
```

This would indicate that the event export task runs at 30 minutes after the hour, every day; the topology export task runs on the hour at midnight and noon, and that the trend export runs 40 minutes after the hour at midnight and noon.

The status field (in this case, `ACTIVE`) is used for internal scheduling purposes by Reporting. `ACTIVE` indicates that a task will be run according to the schedule. `SUSPENDED` indicates that it will not be run even when it becomes due. `LOCK` indicates that the task is either currently running or is in use by one of the scheduler components.

NOTE Modifying the schedule or status for tasks managed by `ovrequestd` is not supported.

For more information, see the `ovrequestd` and `request_list` online reference pages.

B Maintaining a Database

This appendix discusses tasks you need to perform occasionally to maintain the database such as:

- Tuning databases
- Handling log files
- Starting and stopping the database
- Authorizing database access to users
- Improving trend data performance

Database Tuning

This section discusses how to keep HP OpenView and your database operating efficiently by:

- Improving performance
- Creating indexes

Improving Performance

Tuning a database is dependent on parameters such as:

- Machine type
- Memory and swap space utilization
- Disks and controllers
- Number of nodes in your HP OpenView environment
- Applications integrated with HP OpenView
- Additional applications running on the system
- Kernel configuration

Some general suggestions that may improve database performance include the following:

- Use the `nice` facility to increase the priority of a database server and its processes.
- Increase the amount of main memory. If more memory is available, more database queries can be carried out in memory, not on disk, reducing processing time.
- Tune your system's kernel:

HP 9000 Series 700	The number of buffer headers and the number of buffer pages should be set to zero (0). This implies the dynamic management of that resource. Increase the inode cache (1563).

	Be sure the number of open files is sufficient.
HP 9000 Series 800	Increase the number of buffer heads and the number of buffer pages (nbuf=2048, cache=4096 (=16MB)).

- Do not configure swap space on disks storing the database files or transaction log files.
- Be sure you have sufficient free disk space when swap space is configured.

NOTE After increasing the memory for a database facility, carefully monitor the memory utilization using tools such as `monitor` or `glance`. If the processes use too much memory, the swap rate will increase and slow down performance.

Creating Indexes

To facilitate queries that are run often, indexes are created for the tables when the schema are setup. The key column provided in tables allows access to particular rows or sets of rows in tables where data is stored. These keys are used to index the table.

To determine the indexes that are created in the data warehouse, refer to the files listed below. You can find these files in the following directory:

Windows:	`install_dir\conf\analysis\sqlScripts`
UNIX:	`$OV_CONF/analysis/sqlScripts`

- `table_trend.[solid oracle msSqlSrvr]`
- `table_topo.[solid oracle msSqlSrvr]`
- `table_event.[solid oracle msSqlSrvr]`

Handling Log Files

Your relational database may use transaction log files to record changes made to the database or errors that occurred when individual actions failed. These files are essential; in the event of a hardware or software failure, these files provide a way to retrieve the data stored in the database before the system failure.

Because these files are recording network activity, the file size can be large. Check your database documentation for suggestions on managing these files.

For Oracle Relational Databases

Oracle uses a redo log file to record any changes made in an Oracle database. This file is separate from the data files that actually contain the data. There are two parts to the redo log file:

- Online redo log

  The online redo log stores the most current database changes. Once the log file fills up, the file and its contents can be archived as an archived redo log and set apart from the online log. The archiving option can be configured to run automatically in the Oracle database.

- Archived redo log

  Each archived redo log is uniquely identified and stores older data, while allowing the online log to continuously store the most recent database activity.

Procedures for viewing, managing log file contents, and recovering database environments with Oracle log files are described in your Oracle documentation.

For Microsoft SQL Databases

Schedule regular transaction log consolidation backups and full database backups.

Because the database does not need up-to-the-minute recovery, you can set the "`truncate log on checkpoint`" option `on`. This prevents your transaction logs from growing too large, but only allows for recovery up to your last full database backup. Refer to your MS SQL Server documentation for specific procedures.

Starting and Stopping the Database

When you use Oracle or Microsoft SQL Server as the data warehouse database:

1. `ovstart ovdbcheck` validates whether a connection between the data warehouse and the database is successful.

   If the connection is not successful, `ovdbcheck` returns an error.

   `ovstart` does not start Oracle or Microsoft SQL Server. These products should be configured to run at startup.

2. After the database is started by `ovspmd`, `ovdb` attempts to connect to the database once every five minutes.

   If the database is stopped or terminates unexpectedly, `ovdbcheck` reports the error back to `ovspmd`.

Embedded Database

The embedded database starts and stops when NNM starts and stops (that is, with the `ovstart` and `ovstop` commands).

`ovdbcheck` checks whether the embedded database is running. If the database is not running, `ovdbcheck` attempts to start it.

Oracle Database

Use either the Oracle Installer's `dbstart` and `dbshut` utilities, or the `svrmgrl` administration tool. Refer to the Oracle documentation for more help on these tools.

Microsoft SQL Server Database

Use the Microsoft `SQL Enterprise Manager` to start and stop Microsoft SQL Server. To set up Microsoft SQL Server to start when the Windows operating system is started:

1. Start `SQL Enterprise Manager`.
2. In the `SQL Enterprise Manager Configurations` dialog box, select the auto-start options.

Authorizing User Access to the Database

A relational database has a defined set of valid users who can access it. A valid user accesses the database by invoking a database application (such as SQL*Plus), then connects by specifying a user name already defined in the database. Once connected, a user can access more specific types of data in the database according to the access rights specified for the username, as established by a systems or security administrator. More specific information about authorizing user access is described below.

There are two methods that allow you to use the data warehouse tools with other database users:

- Specify the `-u` and `-password` options on the data warehouse tools.
- Change the data warehouse configuration using `ovdwconfig.ovpl`. Refer to the *ovdwconfig.ovpl* (1) manpage or reference page.

NNM creates the default data warehouse user, `ovdb`, at install time. To create additional users, follow the instructions below:

1. Create the user according to your vendor documentation.
2. To validate whether this user is correctly authorized, run:

`ovdwquery -u` ***`newuser`*** **`-password`** ***`password`***

- If the user is correctly authorized, the normal `ovdwquery` prompt displays:

  ```
  Connected to ODBC Data source: tcpip 2690 as NEWUSER.
  Enter SQL command, terminated by ";" ("quit;" to
  terminate):
  ```

- If the user or password are incorrect, an error message displays (similar to the message below):

  ```
  Connect to ODBC data Source "tcpip 2690" failed

  ODBC Error:

  SQLSTATE = 28000

  NATIVE ERROR = 14505
  ```

```
[Solid][SOLID ODBC Driver][SOLID Server]SOLID Server
Error 14505: Connect failed ,

illegal user name or password
```

Oracle Database

To modify an Oracle database's set of current, valid users, use the Oracle Server Manager utilities. These utilities let you add, modify, or delete user accounts in the HP OpenView database. See your Oracle documentation for more information about these utilities.

Users other than `ovdb` who need to access the database tables need to be added.

Changing the Default User and Password for the Data Warehouse

Use the configuration tool, `ovdwconfig.ovpl`, to change the default database user that the data warehouse tools use. Enter the command:

`ovdwconfig.ovpl -type` ***`dbtype`*** **`-u`** ***`user`*** **`-password`** ***`password`***

The specified password is encrypted and stored in *`install_dir`*`\conf\analysis\ovdw.conf` (for Windows) or `$OV_CONF/analysis/ovdw.conf` (for UNIX).

Improving Trend Data Performance

Trend data can grow quickly and consume significant amounts of disk space and CPU resources. You can minimize the impact of large trend data sets by reducing the time it takes to both export and query the data. Some strategies are described below:

- Export the trend data as reduced data instead of raw data.
- If you are exporting reduced data (the default), increase the data reduction interval by using the `ovdwtrend -exportinterval` option. Specify the reduction interval in minutes.
- Increase the amount of data exported with each execution of `ovdwtrend` to minimize the number of export operations. For example, increase the time between export operations. Each export operation includes trend dataset ID lookups and checks for changes in hostnames and topology IDs.
- For an Oracle database, place the `OV_DATA` and `OV_INDEX` tablespaces on different disks. You can specify the location of each of these tablespaces when you run `ovdbsetup`.

  An alternative strategy would be to move one of these tablespaces to a different disk after the initial configuration by using the Oracle SQL command, `ALTER TABLESPACE`, with the `RENAME DATAFILE` option. (See your Oracle documentation for more information.)

C NNM Data Warehouse and Remote Database Connectivity

The default configuration of NNM's data warehouse to an external database assumes a database installed on the same machine as NNM. While this configuration is often adequate, there are circumstances where separating NNM application from the database server is desirable. This appendix shows how to achieve this type of distributed implementation in various environments.

The basis for the connectivity from NNM to the database servers is the use of the network client tools provided by the Oracle products. This takes advantage of the built-in, easy-to-use capabilities of the database products supporting an external data warehouse for NNM.

This appendix does not cover issues related to network configurations, firewall, or security issues. It is a step-by-step guide to use the NNM configuration tools to successfully allow NNM to use a database residing on a separate machine. It addresses only new installations of NNM or those where the data warehouse has been using the embedded database and now is being moved to an external database, specifically Oracle.

SQL Server 2000 as a Remote Database

This scenario involves only computers running Windows, as SQL Server is only available on that platform.

On the Windows machine to be the database server

Install SQL Server 2000 as documented in "Installing and Configuring Microsoft SQL Server" on page 68. Only do the "Configuring Microsoft SQL Server". The ODBC and data warehouse configurations will take place on the database client machine.

The following information assumes that, as specified in the configuration section, there is a database created named `openview` with a user named `ovdb` with the appropriate permissions on the `openview` database.

On the Windows machine on which NNM will reside (database client):

1. Install the SQL Server client utilities.
   a. From the SQL Server installation CD-ROM, select `SQL Server 2000 Components`.
   b. Select `Install Database Server`.
   c. Select `Create a new Instance of SQL Server or install Client Tools`.
   d. At the installation definitions screen choose the following installation: `Client Tools Only`.
2. Open your SQL Enterprise Manager on the database client system. Right click on SQL server group and select New SQL Server Registration.
   a. At the Welcome to the Register SQL Server Wizard screen – click `Next`.
   b. Within the Server screen highlight the name of your database server from the "Available Servers" and click the `Add` button, and click `Next`.

c. Within the Select an Authentication Mode Screen – Make sure that the screen has SQL Server login as the authentication mode, the click `Next`.

d. Within the Select Connection Option screen, enter your login information (for example, user name – `ovdb` – with the `openview` database user's password) and click `Next`.

e. Within the Select SQL Server Group screen – the screen should default to "Add the SQL Server(s) to an existing SQL Server Group" – make sure it shows "SQL Server Group" in the drop down box and click `Next`.

f. Within the Completing the Register SQL Server Wizard – Click `Finish`.

g. If everything was successful you are done with the connection (this will test your connection). Click `Close` to be returned to your Enterprise Manager,

3. Configure the SQL Server ODBC driver to connect to the `openview` database on the server:

a. Click `Start -> Settings -> Control Panel -> Administrative Tools -> Data Sources (ODBC)`

b. Launch the ODBC Administrator Dialog

c. Select the System DSN tab and click `Add`

d. Select the SQL Server driver and click `Finish`

e. Within the "Create a New Data Source to SQL Server", complete the screen. An asterisk prior to the field name implies that you may enter whatever value you choose. Suggested values are supplied to assist you in troubleshooting should that be required. Fields without an asterisk are to be given the designated value.

*Data Source Name	*OVDatabaseServerName*
Description	*OpenView Sql Server ODBC*
Server	*DatabaseServerName* – Choose this from the drop down Choice!

where:

OVDatabaseServerName	A name you choose to identify the data source. Using OV as the first two letters of the data source allows you to identify this as an OpenView repository. Using the hostname of the database server allows a quick determination of the connection. For example, if the database server has been installed on a Windows system with the hostname of `wtest04`, then a useful data source name would be `OVwtest04`.
OpenView Sql Server ODBC	Any description that allows you to know the purpose of this data source.
DatabaseServerName	The fully-qualified host name on which the database server resides. Choose from the drop down choices!

f. After filling in the above information – click Next.

g. At the next popup screen – `Authenticity of the Login ID` – choose "With SQL Server authentication using a login ID and password enter by the user". Enter an appropriate Login ID and Password that you should have created previously on the Database Server system (this example again assumes that there is a database created name `openview` with a user `ovdb` with a password of `ovdb`). If this has not been done you will need to do this prior to completing this step. Click `Next` to continue.

h. At the next popup screen – click on the check box for "Change the default database to" and select the database that you created in the drop down – again this example uses `openview` as the database name. Click `Finish`.

i. At the next popup screen click "Change the language of SQL Server system messages to" choose your preferred language. Click `Finish`.

j. The next popup screen will inform you that "A new ODBC data source will be created with the following configuration". Click `Test Data Source`. The test should complete successfully. Click `OK`.

k. A successful test will return a message stating that the connection is good. If the connection is not established, recheck that the values entered in the Server field of the ODBC configuration matches the value entered

l. Click `OK` twice more to exit the ODBC Data Source Administrator Dialog.

4. If NNM is not installed, please install NNM from your CD-ROM.

5. After NNM is installed, run the data warehouse configuration script from the command line as shown:

```
ovdwconfig.ovpl -u ovdb -password ovdbPassword -type msSqlSrvr -rdb
DataSourceName  -load
```

where:

drive	identifies the drive on which NNM was installed
ovdb	`openview` database user
obdbPassword	the `openview` database user's password into the database
msSqlSrvr	this must be entered exactly as shown
DataSourceName	must match the Data Source Name in the ODBC driver configuration

If the database client software and ODBC driver is configured correctly, the `ovdwconfig.ovpl` script will connect to the `openview` database residing on the server and created the schema necessary to support the data warehouse.

If the configuration is not correct, it is unlikely that you will be able to execute `ovdwconfig` successfully. If this is case, please review the above steps carefully, especially the name used for the data source name in the ODBC driver configuration, the exact value "msSqlSrvr" for the type parameter, and verify the database user name and password.

6. Once `ovdwconfig` has completed successfully, verify that NNM is using the database server for the data warehouse. From the command line, execute:

   <*drive*>:\openview\bin\ovstatus -c ovdbcheck

   The results should show the following:

Name	Pid	State	Last Message
ovdbcheck	453	Running	Connect to ODBC datasource:*DataSourceName*

   NNM is successfully connected to the database server for data warehouse and reporting.

Oracle 8.1.7 on Windows as a Remote Database

NOTE At this writing, Oracle support for version 8.1.7 was expected to end December 31, 2003.

This scenario assumes that the database client on which NNM will run is also using the Windows operating system.

You should use the same version of Oracle between the database server and database client.

On the Windows machine to be the database server:

Install the Oracle 8.1.7 database server as documented in the “Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems)” on page 64.

The following information assumes that, as specified in the configuration section, there is a database created named `openview` with a user named `ovdb` with connect and resource permissions on the `openview` database.

On the Windows machine on which NNM will reside (database client):

1. Install the Oracle 8.1.7 (matching the same version of the Database server) client software.

   a. From the Oracle8.1.7 Client dialog box, select `Custom Installation`. Given the product installation options, select all the default selected components. Also select the following:

      - Oracle Utilities 8.1.7.0.0 –> Oracle Database Utilities 8.1.7.0.0
      - Oracle Enterprise Manager Products 8.1.7.0.0 –> Oracle Enterprise Manger Client 2.2.0.0.0 –> Oracle Enterprise Manager Integrated Applications 2.2.0.0.0 –> Oracle DBA Management Pack 2.2.0.0.0 –>

— Oracle Schema Manager 2.2.0.0.0

— Oracle Storage Manager 2.2.0.0.0

— Oracle Security Manager 2.2.0.0.0

— Oracle Instance Manager 2.2.0.0.0

— SQL Plus Worksheet Oracle

— Oracle DBA Studio

- Oracle Installation Products 8.1.7.0.0 –> Oracle Universal Installer 1.7.1.9.0

b. Click `Next` to continue.

2. If you configured Oracle's Net8 during installation as outlined above, skip to the next step.

   If not, you can manually configure this after the installation as follows:

   a. For Oracle 8.1.7: Within Oracle's program group, select the "Network Administration" and launch the "Net8 Configuration Assistant

   b. Select "Local Net Service Name Configuration:, then click `Next`.

   c. At the next screen, select "Add" then click Next.

   d. At the next screen, select "Oracle 8i database or service" then click `Next`.

   e. At the next screen, enter the name of your global database's name (for example, `openview`), then click Next.

   f. At the next screen, TCP should be the default in the selected protocols windows. Click `Next`.

   g. At the next screen, enter the Host Name of the Oracle server that you will be connecting to (for example, `wtest04.cnd.hp.com`). The radio button "Use the standard port number of 1521" should be selected by default; if not select it and then click `Next`.

   h. At the next screen, click "Yes, perform a test" to test your connectivity to the remote database. (If your connectivity fails at the next screen, you can change the login information with a valid account). Click `Next`.

i. The next screen should display “Connecting… Test Successful”. Click `Next`.

j. At the next screen, accept the defaulted Net Service Name. If this is blank, please set it to the name of your database or service, such as `openview`.

k. At the next screen, accept the default of “No” to configure another net service name at this time. Click `Next`.

l. At the next screen, click `Next`.

m. At the next screen, click `Next`.

3. Configure the Oracle ODBC driver.

a. Launch the ODBC Administrator Dialog (`Settings->Control Panel->Administrative tools->Data Sources (ODBC)`).

b. Select the System DSN tab and click the `Add` button.

c. Click on the Oracle ODBC driver and then the `Finish` button.

d. Complete the screen. An asterisk prior to the field name implies that you may enter whatever value you choose. Suggested values are supplied to assist you in troubleshooting should that be required. Fields without an asterisk are to be given the designated value.

*Data Source Name: *OVDatabaseServerName*

*Description: *Openview Oracle ODBC connection*

*Data Source:

Service Name:	“your Remote Host Name”
User Id:	*ovdb*

where:

OVDatabaseServerName

A name you choose to identify the data source. Using OV as the first two letters of the data source allows you to identify this as an OpenView repository. Using the hostname of the database server allows a quick determination of the connection. For example, if the database server has been installed on a machine running Windows with the hostname of `prod1`, then a useful data source name would be `OVprod1`.

Openview Oracle ODBC connection	A description that allows you to know the purpose of this data source.
Service Name	This must be the fully qualified name of the remote host.
UserId	The Oracle user of the `openview` database per the "Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems)" section.

e. Click `OK` to complete the configuration of the ODBC driver.

4. Configuring Oracle's Network Assistant

   a. Open (from within Oracle's Program group) Oracle's Network Administration: Net8 Assistant

   b. In the navigation section, click on the "+" next to Local

   c. Then click on the "+" next to Service Naming. If the name shown is different than your remote host name, highlight the name and then click on `Edit` and select rename and rename it to the name of your remote host.

   d. Make certain that the "service name" shown is the same as your database's name and that the "Host Name" shown is the same as the your remote host name.

   e. Click `Command` and select `Test Service`. If this test fails, change the username and password to a valid account.

   f. Click `File->Save Network Configuration`.

5. Testing your ODBC connection

   a. Open (from within Oracle's Program group) Oracle's Network - Microsoft's ODBC Administrator

b. Click the `System DSN` tab, and select the system data source that you created in step 3.

c. Click the `Configure` button. This should popup the “Oracle 8 ODBC Driver Configuration”.

d. Click the “Test Connection” button.

e. Inside the “Oracle ODBC Driver Connect” popup, enter (without quotes) the service name, user name (“`ovdb`”), and Password (“`ovdb`”), or your appropriate user information) . Then click the `OK` button.

f. If your connection is successful you will get a small “Testing Connection” popup that shows that your connection was successful. If your connection was unsuccessful you will get an error popup window that should display an Oracle error you can use for diagnosing your connection problems. Your Oracle database administrator should be able to help with any connection issues.

g. You may want to reboot your client system after this step.

6. If NNM is not installed on your Windows machine, install it now.

7. Run the data warehouse configuration script from the command line as shown (enter on a single line):

```
<drive>:\openview\bin\ovdwconfig.ovpl -u ovdb -password ovdbPassword -type
oracle -rdb DataSourceName -load
```

where:

drive	Identifies the drive on which NNM was installed
ovdb	Openview database user
ovdbPassword	The openview database user’s password into the database
`oracle`	This must be entered exactly as shown
DataSourceName	Must match the Data Source Name in the ODBC driver configuration

If the database client software and ODBC driver is configured correctly, the `ovdwconfig.ovpl` script will connect to the openview database residing on the server and created the schema necessary to support the data warehouse.

If the configuration is not correct, it is unlikely that you will be able to execute `ovdwconfig.ovpl` successfully. In this case, please review the above steps carefully, especially the name used for the data source name in the ODBC driver configuration, the exact value "`oracle`" for the type parameter, and verify the database user name and password.

8. Once `ovdwconfig.ovpl` has completed successfully, verify that NNM is using the database server for the data warehouse. From the command line, execute:

   ```
   <drive>:\openview\bin\ovstatus -c ovdbcheck
   ```

   The results should show the following:

Name	Pid	State	Last Message
ovdbcheck	453	Running	Connect to ODBC datasource:*DataSourceName*

NNM is successfully connected to the database server for data warehouse and reporting.

Oracle 9.2.0.1 on Windows as a Remote Database

This scenario assumes that the database client and database server both use the Windows operating system.

NOTE You should use the same version of Oracle between the database server and database client.

On the Windows machine to be the database server:

Install the Oracle 9.2.0.1database server as described in “Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems)” on page 64.

The following information assumes that, as specified in the configuration section, there is a database created named `openview` with a user named `ovdb` with connect and resource permissions on the openview database.

On the Windows machine on which NNM will reside (database client):

1. For Oracle 9.2.0.1, choose `Administrator` at the Installation Types and accept all the defaults.

   a. When the Configuration Tools appear, and within the Net configuration assistant, under Directory Service Access select “No, I want to defer directory service access configuration to another time.” Click `Next`.

   b. In the Net Configuration Assistant: Naming Methods Configuration, Select Naming Methods, choose “`Local`” as the Selected Naming Methods, then click `Next`.

   c. In the next screen (Net Configuration Assistant: Net Service Name Configuration, Database Version), select “`Oracle 8i database or service`” then click `Next`.

d. In the next screen (Net Configuration Assistant: Net Service Name Configuration, Service Name) enter the name of your global database's name, for example `openview`, then click `Next`.

e. In the next screen (Net Configuration Assistant: Net Service Name Configuration, Selected Protocols), TCP should be the default in the selected protocols windows. Click `Next`.

f. In the next screen (Net Configuration Assistant: Net Service Name Configuration, TCP/IP Protocol), you will need to enter the host name of the Oracle Server that you will be connecting to. The radio button " Use the standard port number of 1521" should be selected by default; if not please select it. Then click `Next`.

g. In the next screen (Net Configuration Assistant: Net Service Name Configuration, Test), click `Yes`. Then test your connectivity to the remote database. (It is possible that your connectivity could fail at the next screen. If it does you can change the login information with a valid account information to retest the connection). Click `Next`.

h. The next screen (Net Configuration Assistant: Net Service Name Configuration, Connecting) should display the message "`Connecting … Test Successful`". Click `Next`.

i. At the next screen (Net Configuration Assistant: Net Service Name Configuration, Net Service Name) accept the default Net Service Name. If it is blank, set it to the name of your database or service. Examples in this appendix use `openview`. Click `Next`.

j. At the next screen (Net Configuration Assistant: Net Service Name Configuration, Another Net Service Name?), accept the default of `No`, so that you don't configure another net service name at this time. Click `Next`.

k. At the next screen (Net Configuration Assistant: Naming Methods Configuration Done), click `Next`.

l. At the next screen (Net Configuration Assistant: Done), click `Finish`.

m. Click `Next` to continue.

Once the Oracle client is installed, you might need to reboot your system.

2. If you configured Net Configuration during the installation, skip this step and continue with the next step in this process

However, if you didn't configure Oracle's Net Configuration Assistant as outlined above during the installation, you can manually configure this after the installation as follows:

a. For Oracle 9.2.0.1: Within Oracle's program group, select the "Configuration and Migration Tools " and launch the "Net Configuration Assistant

b. Select "Local Net Service Name Configuration", then click `Next`.

c. At the next screen, select `Add` then click `Next`.

d. At the next screen, select `Oracle 8i database or service` then click `Next`.

e. At the next screen, enter the name of your global database's name (for example, `openview`), then click `Next`.

f. At the next screen, TCP should be the default in the selected protocols windows. Click `Next`.

g. At the next screen, enter the host name of the Oracle server that you will be connecting to (for example, `wtest04.cnd.hp.com`. The radio button "Use the standard port number of 1521" should be selected by default; if not, select it. Then click `Next`.

h. At the next screen, click "Yes, perform a test" to test your connectivity to the remote database. If your connectivity fails at the next screen, you can change the login information with a valid account. Click `Next`.

i. The next screen should display the message "`Connecting... Test Successful`". Click `Next`.

j. At the next screen, accept the defaulted Net Service Name. If it is blank, set it to the name of your database or service – e.g., `openview`.

k. At the next screen - accept the default of "No" to configure another net service name at this time. Click Next.

l. At the next screen, click `Next`.

m. At the next screen, click `Next`.

3. Configure the Oracle ODBC driver.

a. Launch the ODBC Administrator Dialog (`Settings->Control Panel->Administrative Tools->Data Sources (ODBC)`

b. Select the System DSN tab and click the `Add` button.

c. Click on the Oracle 9 ODBC driver and then the `Finish` button.

d. Complete the screen. An asterisk prior to the field name implies that you may enter whatever value you choose. Suggested values are supplied to assist you in troubleshooting should that be required. Fields without an asterisk are to be given the designated value.

*Data Source Name:	*OVDatabaseServerName*
*Description:	*Openview Oracle ODBC connection*
*Data Source:	*OVORACLE* - this can be whatever you want to call it.
*User Id:	*ovdb*

where:

OVDatabaseServerName

A name you choose to identify the data source. Using OV as the first two letters of the data source allows you to identify this as an OpenView repository. Using the hostname of the database server allows a quick determination of the connection. For example, if the database server has been installed on an NT box with the hostname of "prod1", then a useful data source name would be "OVprod1".

Openview Oracle ODBC connection

A description that tells you the purpose of this data source.

UserId

The Oracle user of the `openview` database, as described in "Installing and Configuring an Oracle Relational Database (on Windows® Operating Systems)" on page 64

4. Testing your ODBC connection

a. Click the "Test Connection" button.

b. Inside the "Oracle ODBC Driver Connect" popup, enter (without quotes) the service name, remote host name, user name (e.g., `ovdb`), and password (e.g., `ovdb`), or your appropriate user information. Then click `OK`.

c. If your connection is successful you will get a small "Testing Connection" popup that shows that your connection was successful.

d. If your connection was unsuccessful you will get an error popup window that should display an Oracle error you can use for diagnosing your connection problems – your Oracle DBA should be able to help with any connection issues. The most common problem is the user id and password are set to something like "Scott" and "Tiger" as the password – change these to appropriate account information.

e. Click OK to complete the configuration of the ODBC driver.

5. Check the Net Service Name

   a. From within Oracle's Program group, open Oracle's Configuration and Migration Tools: Net Manager

   b. In the navigation area, click on the "+" next to `Local`

   c. Click on the "+" next to `Service Naming`. If the name shown is different than your remote host name, highlight the name, click `Edit` and select `Rename` to give it the name of your remote host.

   d. Make certain that the `Service Name` shown is the same as your database's name and that the "Host Name" shown is the same as the your remote host name.

   e. Click `File–>Save Network Configuration`.

   f. You may want to reboot your client system at this time.

6. If NNM is not installed, install it now.

7. After NNM is installed, run the data warehouse configuration script from the command line as shown (enter on a single line):

```
<drive>:\openview\bin\ovdwconfig.ovpl -u ovdb -password ovdbPassword -type
oracle -rdb DataSourceName -load
```

where:

drive	Identifies the drive on which NNM was installed

ovdb	Openview database user
ovdbPassword	The openview database user's password into the database
`oracle`	Must be entered exactly as shown
DataSourceName	Must match the Data Source Name in the ODBC driver configuration

If the database client software and ODBC driver is configured correctly, the `ovdwconfig.ovpl` script will connect to the `openview` database residing on the server and created the schema necessary to support the data warehouse.

If the configuration is not correct, it is unlikely that you will be able to execute `ovdwconfig.ovpl` successfully. In this case, review the above steps carefully, especially the name used for the data source name in the ODBC driver configuration, the exact value `oracle` for the type parameter, and verify the database user name and password.

8. Once `ovdwconfig.ovpl` has completed successfully, verify that NNM is using the database server for the data warehouse. From the command line, execute:

   <*drive*>:\openview\bin\ovstatus -c ovdbcheck

   The results should show the following:

Name	Pid	State	Last Message
ovdbcheck	453	Running	Connect to ODBC datasource:*`DataSourceName`*

NNM is successfully connected to the database server for data warehouse and reporting.

Oracle on UNIX as a Remote Database

This scenario is used when the Oracle database server resides on a UNIX box. NNM will be installed on a computer running Windows You must have the NNM installation CD-ROMs for both the Windows and UNIX platforms.

This configuration uses parts of the UNIX data warehouse setup with parts of the Windows setup. Some scripts run on the database server will return errors, because not all of the expected executables will be present. These “errors” are acceptable and will not interfere with NNM on Windows using an Oracle database residing on a computer running UNIX.

IMPORTANT This section identifies what “errors” to expect.

On the UNIX machine to be the database server

1. Install the Oracle8.1.7 or Oracle 9.2.0.1 database server as described in “Installing and Configuring an Oracle Relational Database (on UNIX Systems)” on page 50.

NOTE Only the first section, titled “Installing Oracle,” should be completed at this time.

2. Create the directory for the setup scripts. The scripts must reside in `/opt/OV/bin`:

   ```
   prod1> mkdir -p /opt/OV/bin
   ```

3. Browse the NNM installation CD-ROM for your UNIX platform. From the `/bin` directory, copy the `ovdbsetup` scripts to the local directory:

   ```
   prod1> cp ovdbsetup* /opt/OV/bin/.
   ```

4. Configure the Oracle database by launching the OpenView database setup script. You must be “root” to run the ovdbsetup:

```
prod1> ovdbsetup -o
```

This will return:

```
Entering Phase 1 (ovdbsetupo1.sh)
```

You will need to answer a series of questions identifying the defaults to be used when creating the openview database. This phase should complete successfully.

NOTE Ensure that the paths you specify for the data and index directories have sufficient disk space before proceeding. Insufficient disk space causes `ovdbsetup` to fail. Please read the warning in "Configuring Oracle with ovdbsetup" on page 52 of the "Installing and Configuring an Oracle Relational Database (on UNIX Systems)" section.

Once the database has been created you be informed that the next script has started:

```
Entering Phase 2 (ovdbsetupo2.sh)
```

After supplying the password for the Oracle user, "`system`", the script continues. This phase should also complete successfully and launch the third phase:

```
Entering Phase 3 (ovdbsetupo3.sh)
```

After supplying responses concerning the export of topology and SNMP trend data, a great deal of activity will appear on the screen.

IMPORTANT This phase will report errors.

Here is what to expect:

- Output to the screen will show that new Oracle network files were created. The script will attempt to connect to the Listener but because the Listener has not yet been started, the attempts will fail. This is expected behavior.

  Typically, NNM relinks the shared library so as to include its own library files. Since the NNM executable will reside on another machine, it is not necessary to relink the Oracle shared library. This is acceptable.

- The script will be unable to relink the shared library producing the following message:

  ```
  Unable to regenerate libclntsh.sl: clntsh.mk not found
  ```

  The name of the file for which the regeneration is being attempted is specific to your platform. The `libclntsh.sl` in the above message is an example from HP-UX 10.20, but other versions produce a similar message.

- The output should now reflect the Listener being started. The following line indicates that the Listener has been launched:

  ```
  The command completed successfully.
  ```

- There will be several messages to the following effect:

  ```
  /opt/OV/bin/ovdbsetupo3.sh[nnn]:
  /opt/OV/bin/ovdwcfggetset: not found
  ```

  The `ovdwcfggetset` normally writes database configuration information for NNM. Since NNM will reside on the database client, the configuration information will be created on the client as described later in this process. These messages can be ignored.

- A set of messages will state:

  ```
  Warnings detected while creating SQL tables using…
  ```

  Again, these messages can be ignored since the schema creation will be handled from the database client.

- A final set of errors will be communicated to the screen:

```
/opt/OV/bin/ovdbsetupo3.sh[nnn]: /opt/OV/bin/ovstop: not found
/opt/OV/bin/ovdbsetupo3.sh[nnn]: /opt/OV/bin/ovdwcfggetset: not found
/opt/OV/bin/ovdbsetupo3.sh[nnn]: /opt/OV/bin/ovstart: not found
```

  These messages are referring to components that will exist on the client. These messages are expected in this configuration.

- A final message will appear stating:

  ```
  ovdbsetupo3.sh phase for NNM configuration completed
  ```

The database server configuration is complete.

On the Windows machine on which NNM will reside (database client)

Based on the version of Oracle installed on the server, go to the section in this appendix titled "Oracle v.sv.x on Windows as a Remote Database". Locate the section titled "On the Windows machine on which NNM will reside (database client)". Follow the instructions for the client configuration.

Remote Data Warehouse configuration for NNM and Oracle residing on UNIX systems of different types

This scenario involves an HP-UX to Solaris configuration.

IMPORTANT HP recommends you use the same version of Oracle on both the database server and database client.

1. The host where you install NNM is designated as the database client system for the rest of this discussion. On that host, install NNM in the usual way.
2. On the system to be the database server, install the Oracle database server as documented in the "Installing and Configuring an Oracle Relational Database (on UNIX Systems)" on page 50.

NOTE The following information assumes that, as specified in the configuration section, there is a database created named `openview` with a user named `ovdb` with connect and resource permissions on the openview database.

3. On the system on which NNM will reside (database client), install the Oracle client software (matching the same version of the Database server). From the Oracle Client dialog box, select Custom Installation. Accept all the default selected components.
4. Export the OV directories so the database server can mount them.
   - If NNM is on HP-UX:
     a. In the `/etc/exports` file root needs to be given write access to the mounted directories. Add the following lines as shown:

```
/opt/OV root=DatabaseServerHostName

/etc/opt/OV -root=DatabaseServerHostName

/var/opt/OV -root=DatabaseServerHostName
```

b. From the command line, run: `exportfs -av`

- If NNM is on Solaris:

a. In the `/etc/dfs/dfstab` file add the following lines as shown:

```
share -o root=DatabaseServerHostName /opt/OV

share -o root=DatabaseServerHostName /etc/opt/OV

share -o root=DatabaseServerHostName /var/opt/OV
```

b. If the `dfstab` file only had commented lines in it prior to adding the "share" lines above, you will need to start the nfs server before the directories can be mounted by the database server:

`sh /etc/init.d/nfs.server start`

- If NNM is on Linux:

a. In the `/etc/exports` file, add the following lines as shown:

```
/opt/OV DatabaseServerName(rw,no_root_squash)

/etc/opt/OV DatabaseServerName(rw,no_root_squash)

/var/opt/OV DatabaseServerName(rw,no_root_squash)
```

b. From the command line, execute: `exportfs -av`

5. Mount the OV directories from the NNM server with NFS to the database server.

```
umask 022
mkdir /opt/OV /etc/opt/OV /var/opt/OV
mount nnm_server:/opt/OV   /opt/OV
mount nnm_server:/etc/opt/OV   /etc/opt/OV
mount nnm_server:/var/opt/OV   /var/opt/OV
```

6. From the database server, configure the Oracle database by launching the OpenView database setup script. You must be `root` to run the `ovdbsetup.sh` script:

`/opt/OV/bin/ovdbsetup -o`

This will return:

```
Entering Phase 1 (ovdbsetupo1.sh)
```

You will need to answer a series of questions identifying the defaults to be used when creating the openview database. This phase should complete successfully.

NOTE Ensure that the path you specify for the data and index directories have sufficient disk space before proceeding. *Insufficient disk space causes `ovdbsetup` to fail.*

Phases 1 and 2 should complete successfully.

Phase 3 should fail with the following (or similar) error:

```
Regenerating libclntsh.so for Oracle
/opt/OV/iodbc/src/oracle: does not exist
```

Phase 3 (`ovdbsetupo3.sh`) attempts to relink the Oracle shared library and place the file in `/opt/OV/iodbc/lib`, but on a cross-platform implementation, the shared libraries are not the same nor located in the same place for HP-UX as for Solaris.

7. Copy the Oracle shared library and put it where NNM expects it.

   On the NNM Server, go to the Oracle library directory. Find the shared library files:

   ```
   cd $ORACLE_HOME/lib
   ls -la libclntsh*
   ```

   On an HP-UX machine, the file should be `libclntsh.sl`. On Solaris, it is `libclntsh.so`. Sometimes there is a link from the shared library file to the actual file, like `libclntsh.sl.1.0`. Copy the file with the bits (not the link) from this directory to the NNM library directory renaming the file as needed to either `libclntsh.sl` (for HP-UX) or `libclntsh.so` (for Solaris and Linux):

```
cp libclntsh.goodfile /opt/OV/iodbc/lib/libclntsh.correct_file_extension
```

   Make sure that the file you just copied to `/opt/OV/iodbc/lib` is owned by root:

```
chown root  /opt/OV/iodbc/lib/libclntsh.correct_file_extension
```

   Check that the `lib` directory is owned by root:sys and that everybody has execute privileges.

```
chown bin:bin /opt/OV/iodbc/lib
chmod +x  /opt/OV/iodbc/lib
```

Make sure the Oracle shared library has the required execute permissions.

- On HP-UX:

  ```
  chmod +x  $ORACLE_HOME/lib/libclntsh.sl
  chmod +x /opt/OV/iodbc/lib/libclntsh.sl
  ```

- On Solaris and Linux:

  ```
  chmod +x  $ORACLE_HOME/lib/libclntsh.so
  chmod +x /opt/OV/iodbc/lib/libclntsh.so
  ```

8. Change ownership of database configuration files, if needed, on the NNM server.

   Make sure that the NNM database configuration files have the required ownership. If the exported file systems did not allow root user access, the files created over the mounted file systems will show user and group as `nobody:nogroup`. If `/etc/opt/OV/share/conf/ovdbconf` is not owned by root, make the following change:

   ```
   chown root:sys /etc/opt/OV/share/conf/ovdbconf
   chown root:sys /opt/OV/iodbc/lib/libclntsh.*
   ```

9. Copy the Oracle network files from the database server to the NNM server. These files must reside on both systems in the right locations for that OS.

   If you are copying Oracle network files from an HP-UX database server to an NNM server on Solaris, then from the database server do the following commands:

   ftp ***nnm_server*** use **root**'s login for username/password
   ```
   cd /var/opt/oracle
   lcd /etc
   prompt off
   mput *.ora
   ```

NOTE If you are copying to a Solaris box, you may need to create a `/var/opt/oracle` directory.

If you are copying Oracle network files from a Solaris database server to an NNM server on HP-UX then from the database server copy:

`ftp` ***`nnm_server`*** use root's login for username and password

```
cd /etc
lcd /var/opt/oracle
prompt off
mput *.ora
```

This should copy the following files from the database server to the NNM server:

```
listener.ora
sqlnet.ora
tnsnames.ora
```

NOTE This process assumes that the value for ORACLE_HOME is the same on both systems! If the value is different, the `.ora` files will need to be modified accordingly

10. Create or change the appropriate values in the data warehouse configuration files (`ovdwenvs.conf` and `ovdw.conf`)

NOTE Because the program used to set these values is compiled, you can not run the HP-UX version on Solaris, nor the Solaris version on HP-UX.

All of the following commands must be executed on the NNM server regardless of the specific operating system:

```
export DW_ENVIRS=/etc/opt/OV/share/conf/analysis/ovdwenvs.conf
export DW_CONF=/etc/opt/OV/share/conf/analysis/ovdw.conf

/opt/OV/bin/ovdwcfggetset -file $DW_ENVIRS -s "=" \
    -set "ORACLE_HOME=$ORACLE_HOME" \
    >> /var/opt/OV/share/log/externalDbConf.log

/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" -set "user: ovdb" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

```
/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" \
    -set "password: ovdb" -encrypt 1 \
    >> /var/opt/OV/share/log/externalDbConf.log
```

- If the NNM server is an HP-UX machine:

```
/opt/OV/bin/ovdwcfggetset -file $DW_ENVIRS -s "=" \
    -set "SHLIB_PATH=$ORACLE_HOME/lib:/opt/OV/iodbc/lib" \
    >> /var/opt/OV/share/log/externalDbConf.log

/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" -set "dbname: OVoracle" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

- If the NNM server is a Solaris machine:

```
/opt/OV/bin/ovdwcfggetset -file $DW_ENVIRS -s "=" \
    -set "LD_LIBRARY_PATH=/opt/OV/iodbc/lib:$ORACLE_HOME/lib" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

- If the NNM server is a Linux machine:

```
/opt/OV/bin/ovdwcfggetset -file $DW_ENVIRS -s "=" \
    -set "LD_LIBRARY_PATH=$ORACLE_HOME/lib:/opt/OV/iodbc/lib" \
    >> /var/opt/OV/share/log/externalDbConf.log

/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" \
    -set "dbname: OVoracle" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

- If the Solaris NNM server is using an Oracle client version of 9.x:

```
/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" -set "dbname: OVoracle9" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

- If the Solaris NNM server is using an Oracle client version of 8.x:

```
/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" -set "dbname: OVoracle8" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

- otherwise, if the Oracle client version is 7.x execute:

```
/opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" -set "dbname: OVoracle" \
    >> /var/opt/OV/share/log/externalDbConf.log
```

11. Verify that the Oracle listener process is running on the database server:

    ```
    ps -ef | grep tnslsnr
    ```

    If the listener process is not running, you will need to start it:

    ```
    su - oracle
    lsnrctl start
    ```

12. Disable the embedded database on the NNM server then reconnect to the brand new remote oracle database instead!

    a. Stop the data warehouse connection to Solid:

    ```
    /opt/OV/bin/ovstop ovdbcheck
    ```

    b. Turn off the embedded database permanently:

    ```
    /opt/OV/bin/ovdwcfggetset -file $DW_CONF -s ": @" \
    -set "db:   no"  >>
    /var/opt/OV/share/log/externalDbConf.log
    ```

    c. Restart `ovdbcheck`:

    ```
    /opt/OV/bin/ovstart ovdbcheck
    ```

    d. Verify that `ovdbcheck` is connecting successfully to the correct database:

    ```
    /opt/OV/bin/ovstatus -c
    ```

    This should return the following in the "Last Message" column for `ovdbcheck`:
    ```
    Connected to ODBC datasource: OVoracle
    ```

13. Verify the connection:

    Run `ovdbcheck` in troubleshooting mode to determine if NNM can successfully connect to the openview database:

    ```
    ovdbcheck -t
    ```

    A successful **ovdbcheck -t** will return:

    ```
    Connecting as user "ovdb" with password
    Validate accessability to "OVoracle" database
    ```

14. Create the tables for the data warehouse.

    This would normally be taken care of in the `ovdbsetupo3.sh` script but because this script fails in a cross-platform implementation, the table creation must be handled somewhat manually. From the NNM server execute the following commands:

    ```
    cd /etc/opt/OV/share/conf/analysis/sqlScripts
    ```

    ```
    /opt/OV/bin/ovdwquery -file tables_trend.oracle  \
    >> /var/opt/OV/share/log/externalDbConf.log 2>&1
    ```

    ```
    /opt/OV/bin/ovdwquery -file tables_topo.oracle  \
    >> /var/opt/OV/share/log/externalDbConf.log 2>&1
    ```

```
/opt/OV/bin/ovdwquery -file tables_event.oracle  \
>> /var/opt/OV/share/log/externalDbConf.log 2>&1
```

```
/opt/OV/bin/ovdwquery -file tables_common.oracle  \
>> /var/opt/OV/share/log/externalDbConf.log 2>&1
```

These scripts append to the `$OV_LOG/externalDbConf.log` file, and should complete with `"Operation Completed"`. Check this file if the scripts do not complete successfully.

15. Use `ovdwquery` on the NNM server to check for the existence of the data warehouse tables:

    `ovdwquery`

    When `ovdwquery` states `"Enter SQL command. . ."` enter:

    `select table_name from user_tables;`

    This will return a list of all tables owned by `ovdb`. If you do not get any tables, verify that you ran each of the 4 "tables" scripts described in the previous step. At this point, the NNM processes should be communicating with the Oracle database on the remote system.

    Scheduling reports will automatically schedule data warehouse exports.

D Reference Pages

Viewing and Printing Manpages

For certain topics in this manual, the names of specific reference (`man`) pages were provided. You can either view these reference pages online or print them.

The reference pages discussed in this manual are listed below:

- *ovcolqsql*(1)
- *ovcoldelsql*(1M)
- *ovcoltosql*(1M)
- *ovdbcheck*(1M)
- *ovdbdebug*(1M) for UNIX only
- *ovdbsetup*(1M)
- *ovtopmd*(1M)
- *ovdwevent*(1)
- *ovdweventflt*(1)
- *ovdwtopo*(1)
- *ovdwtrend*(1)
- *ovdwconfig.ovpl*(1M)
- *ovdwloader*(1M)
- *ovdwunloader*(1M)
- *ovcolmigo*(1M)
- *ovdwquery*(1M)
- *ovdwunloader*(1M)

On Solaris systems, consult your system documentation for information on accessing manpages online or printing them.

On Linux systems, consult your system documentation for information on accessing manpages online or printing them.

On Windows systems, to view and print reference pages from the menu bar, select `Help:NNM->Reference Pages`.

On HP-UX systems, the following procedure is just one suggestion for displaying or printing manpages. This procedure does require that you have manpages installed locally on your system. (If your network provides manpages remotely, such as from a central server, then check with your system administrator on how to access them.)

1. Determine what syntax and options the `man` command is using. At a command-line prompt, type

   `strings /usr/bin/man | grep col`

   You should receive a message similar to the following:

   ```
   tbl -TX %s |neqn|nroff -h -man|col -x > %s
   tbl -TX %s |neqn|nroff -man|col -x|%s
   ```

2. Determine where on your system the manpage files are kept. Type

   `echo $MANPATH`

   A list with one or more directories appears. Multiple directories are separated by colons (for example, `/usr/local/man:/usr/man`). HP OpenView recommends that you check the contents of each directory to make sure it contains manpage files.

3. Use the command syntax shown in Step 1 in one of two ways:

   - Specify the qualified path of the directory containing manpage files as the `%s` value in the command.

     For example, the *bggen*(1) manpage is stored in the `/usr/man/man1` directory on your system, but you are not currently in that directory. To display this manpage online, enter the following command:

     `tbl -TX /usr/man/man1/bggen.1 |neqn|nroff -man|col|more`

     For example, to send the *bggen*(1) manpage to a file for printing, enter the following command:

     `tbl -TX /usr/man/man1/bggen.1 |neqn|nroff -man|col >` /filename

   - `Go` to the directory on your system that contains manpage files. Specify the command name and number as the `%s` value.

     For example, to display the *bggen*(1) manpage online from the `/usr/man/man1` directory, enter the command:

```
tbl -TX bggen.1 |neqn|nroff -man|col| more
```

For example, to send the *bggen*(1) manpage to a file for printing, enter the following command:

```
tbl -TX bggen.1 |neqn|nroff -man|col > /filename
```

Index

A

automatic report generation, 40
availability reports,web, 91

B

backup
  correcting failed for embedded database, 195
  of the embedded database, 195–198
  of the Microsoft SQL Server database, 200
  of the Oracle database, 199
  restoring the embedded database, 197
  using ovdwevent, 201
  using ovdwtrend, 201
backup and recovery methods
  overview of, 193
book contents, 16
Bookmarking, 90

C

common data tables
  nnm_event_data table, 189

D

data collection
  automatic, 89
  automatic,stopping, 97
data collector tools, 30
  ovcoldelsql, 30
  ovcolqsql, 30
  ovcoltosql, 30
  ovdwtrend, 29
data source
  for ODBC, 23
data warehouse
  defined, 19
  overview of tasks in this manual, 18
  supported databases, 20
  textual data, 109
database types
  event database, 26
  map database, 25
  OVW object database, 25
  topology database, 25
  trend database, 26
  used in HP OpenView, 25
database-related processes
  additional reading for, 28
  netmon, 27
  overview of, 26
  ovrequestd, 27
  ovtopmd, 28
  ovwdb, 28
  pmd, 28
  snmpCollect, 28
database-related tools
  list of data collector tools, 30
  nnmRptConfig.exe, 31
  nnmRptPresenter.exe, 31
  ovcoldelsql, 30
  ovcolqsql, 30
  ovcoltosql, 30
  ovdbcheck, 29
  ovdbdebug, 29
  ovdbsetup, 29
  ovdwevent, 30
  ovdwtopo, 30
  ovdwtrend, 29
  overview of, 29
databases
  remote, 49, 237
  supported, 20
deleting web reports, 98
disk space
  example calculation for trend data, 75
  for trend data, 72

E

embedded database, 72
  described, 45
  disabling, 77
  enabling, 77
  starting and stopping, 231
environment variables, 51
event data
  exporting, 118
event data tables
  nnm_event_cat table, 180
  nnm_event_daily table, 187
  nnm_event_detail table, 182
  nnm_event_limits table, 188
  nnm_event_monthly table, 187
  nnm_event_sev table, 180
  nnm_event_thresh table, 185
  nnm_event_varbinds table, 184
  nnm_event_weekly table, 187
  nnm_event_yearly table, 187
event database, 26

exception reports
  configuring, 93
exception reports,web, 91
export utilities
  how to use, 103
  ovdwevent, 105, 112
  ovdwtopo, 105, 107
  ovdwtrend, 105, 108
  running with a job scheduler, 104

F

flow of data
  described, 106

H

HP OpenView, types of databases in, 25

I

inventory reports,web, 92

K

kernel configuration, 227

L

log files, 208, 229

M

manpages
  listed, 270
  procedure for viewing and printing, 270
manual contents, 16
map database, 25
Microsoft SQL Server
  installing, 68–??
Microsoft SQL Server database
  disk space for trend data, 73
  starting and stopping, 231
migrating
  between different databases, 85
  cron jobs, 82
  event data to NNM 6.0, 84
  scheduled tasks, 82
  topology data to NNM 6.0, 83
  trend data to NNM 6.0, 83

N

netmon, 27
nnm_classes table, 145
nnm_event_cat table, 180
nnm_event_daily table, 187
nnm_event_data, 189
nnm_event_detail table, 182
nnm_event_limits table, 188
nnm_event_monthly table, 187
nnm_event_sev table, 180
nnm_event_thresh table, 185
nnm_event_varbinds table, 184
nnm_event_weekly table, 187
nnm_event_yearly table, 187
nnm_interfaces table, 146
nnm_networks table, 150
nnm_nodes table, 151
nnm_nodes_cap table, 160
nnm_objects table, 142
nnm_segments table, 153
nnm_stations table, 156
nnm_topology table, 159

O

ODBC
  data source described, 23
  defined, 20
  setting up for Microsoft SQL Server, 69
  setting up for Oracle for Windows NT, 65
  setting up for Oracle on NT, 65
Open Database Connectivity, 20
operating systems supported, 45
Oracle database
  and NNM AE Extended Topology, 48
  authorizing user access, 233
  creating database indexes, 228
  disk space for trend data, 72
  environment variables, 51
  installing on UNIX, 50–56, 64–??
  orainst installer program, 51
  post-installation script, 52
  redo log file, 229
  starting and stopping, 231
orainst Oracle installer program, 51
ovbackup.ovpl, 196
ovcoldelsql, 30
ovcolmigo, 81
ovcolqsql, 30, 124
ovcoltosql, 30
ovdbcheck, 29, 71, 207, 213
  validating connection to NNM with, 71
ovdbdebug, 29, 71, 205
  validating connection to NNM with, 71
ovdbsetup, 29

using, 55
ovdwconfig.ovpl, 29, 81
ovdwevent, 30, 105, 112
ovdwloader, 81
ovdwquery, 124
ovdwtopo, 30, 105, 107
ovdwtrend, 29, 105, 108, 110
ovdwunloader, 81
ovrequestd, 27
ovtopmd, 28
OVW object database, 25
ovwdb, 28

P

performance reports,web, 92
performing queries, 124–128
on event tables, 126
on topology tables, 125
on trend tables, 127
pmd, 28

R

recovery methods for data. See backup and recovery methods
reference pages for commands. See manpages
relational database
selecting, 45
validating connection to NNM, 71
relational database capabilities
for end-users, 20
relational databases
authorizing user access, 232
handling log files, 229
supported products, 45
tuning of, 227
remote database, 49, 237
removing web reports, 98
report templates,web, 89
reporting
tools, 31
using SQL, 129
web, 22, 89
reports,automatic
configuring, 89
reports,web
availability, 91
configuration URL, 90
configuring, 89, 92
deleting, 98
e-mail, 89
exception, 91
inventory, 92
localized, 99
performance, 92
removing, 98
resuming, 95
stopping, 97
suspending, 95
templates, 91
viewing, 94
restoring
embedded database, 197
resuming web reports, 95

S

scheduling
data export, 117
event data export, 118
topology data export, 118
trend data export, 119
using cron, 118
schema
entity diagram for event data tables, 135
entity diagram for topology data tables, 135
entity diagram for trend data tables, 135
entity diagrams, 135
schemas
described, 122
sending a query
from a file, 125
from standard input, 125
snmp_daily_trend table, 172
snmp_monthly_trend table, 176
snmp_raw_trend table, 169
snmp_reduced_trend table, 170
snmp_trend_dataset table, 167
snmp_trend_descript table, 166
snmp_trend_version table, 165
snmp_weekly_trend table, 174
snmpCollect, 28
space requirements, 72
SQL
features of, 24
stopping web reports, 97
string data, 109
suspending web reports, 95

T

task overview, for this manual, 18

textual data storage, 109
thresholds, 41
topology data
  deleting, 105
  exporting, 118
topology database, 25
topology tables
  data conversions, 139
  nnm_classes table, 145
  nnm_interfaces table, 146
  nnm_networks table, 150
  nnm_nodes table, 151
  nnm_nodes_cap table, 160
  nnm_objects table, 142
  nnm_segments table, 153
  nnm_stations table, 156
  nnm_topology table, 159
  object IDs, 139
trend data, 72, 209, 235
  exporting, 119
  improving performance, 235
trend data tables
  snmp_daily_trend table, 172
  snmp_monthly_trend table, 176
  snmp_raw_trend table, 169
  snmp_reduced_trend table, 170
  snmp_trend_dataset table, 167
  snmp_trend_descript table, 166
  snmp_trend_version table, 165
  snmp_weekly_trend table, 174
trend database, 26
troubleshooting
  avoiding duplicate or overlapping data, 209
  connection problem with Microsoft SQL Server, 218
  connection/configuration problems for Oracle, 216
  database connection problems, 210
  problem starting embedded database, 213
  strategies for trend data problems, 209
  using log files, 208
  using ovdbcheck, 207
  using ovdbdebug, 205

U

universal pathnames
  using, 45

V

viewing web reports, 94

®

invent

Java™
COMPATIBLE