Oracle9*i*AS Personalization

Getting Started with Oracle9*i*AS Personalization

Release 9.0.1

July 2001

Part No. A87535-01

ORACLE

Oracle9*i*AS Personalization Getting Started with Oracle9*i*AS Personalization, Release 9.0.1

Part No. A87535-01

Copyright © 2001 Oracle Corporation. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs, except to the extent required to obtain interoperability with other independently created software or as specified by law, is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle9*i* is a trademark or registered trademark of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments

Oracle9*i*AS Personalization Getting Started with Oracle9*i*AS Personalization, Release 9.0.1

Part No. A87535-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- DARWINDOC@us.oracle.com
- FAX: 781-684-7738 Attn: Oracle9*i*AS Personalization Documentation
- Postal service:
  Oracle Corporation
  Oracle9*i*AS Personalization Documentation
  200 Fifth Avenue
  Waltham, Massachusetts 02451
  U.S.A.

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

Preface

Oracle9*i*AS Personalization (OP) provides real-time personalization for Web sites using an integrated real-time recommendation engine that is embedded in an Oracle database. OP is based on data mining technology and modeling; it builds a predictive model of customer preferences using Web-based behavioral data collected by a Web site as well as demographic data.

This manual is designed to introduce Java programmers and DBAs to the basic components and interfaces of OP.

Intended Audience

This manual is intended for users of Oracle9*i*AS Personalization:

- DBA who administers OP
- REAPI Web applications designer
- REAPI Web applications programmer

Structure

This manual contains the following chapters, an appendix, and a glossary:

- Chapter 1, "Introducing Oracle9iAS Personalization"
- Chapter 2, "The OP Administrative UI"
- Chapter 3, "Using the OP Administrative UI"
- Chapter 4, "Using the REAPI Demo"
- Appendix A, "Recommendation Algorithms"
- Glossary (of terms related to Oracle9*i*AS Personalization)

Prerequisites

The following software is required for the Administrative UI browser:

- Netscape 4.75 or Internet Explorer 5.5 with SP1.

Configuration Details

- Two systems, one loaded with Oracle9*i* and the other with Oracle9*i*AS.
- MOR, MTR, and RE schemas on the same machine (the machine where Oracle9*i* is loaded).
- REAPI Demo operates off of JServ on Oracle9*i*AS. The Administrative UI operates off of the Oracle 9*i* JServ / HTTP Server.

Where to Find More Information

The documentation set for Oracle9*i*AS Personalization at the current release consists of the following documents:

- README.htm, on the Oracle9*i*AS Personalization CD; this file contains platform-specific installation instructions.
- *Oracle9*i*AS Personalization Release Notes,* Release 9.0.1.
- *Oracle9*i*AS Personalization Administrator's Guide*, Release 9.0.1 (includes installation instructions that are the same across all platforms).
- *Getting Started with Oracle9*i*AS Personalization,* Release 9.0.1 (this document).
- *Oracle9*i*AS Personalization Recommendation Engine API Programmer's Guide,* Release 9.0.1. A programmer's manual for programming the recommendation engines in real time.
- *Oracle9*i*AS Personalization Recommendation Engine Batch API Programmer's Guide,* Release 9.0.1. A programmer's manual for obtaining bulk recommendations.

Related Manuals

For more information about the database underlying OP, see:

- *Oracle9*i *Administrator's Guide*
- *Oracle9*i *Application Server Installation Guide* (the appropriate version for your operating system).

Requirements

OP documentation is distributed on the same CD that OP is distributed on. Documentation is provided in PDF and HTML formats.

After OP is installed, the OP documentation can be read by opening the following URL using either Netscape or Internet Explorer:

```
http://server/opDoc/op.901/index.htm
```

where *server* is name of the system where OP is installed.

You can read or print the documentation directly from the CD or from your browser.

To view the PDF files, you will need

- Adobe Acrobat Reader 3.0 or later, which you can download from `www.adobe.com`.

To view the HTML files, you will need

- Netscape 4.x or later, or
- Internet Explorer 4.x or later

Online Help

Oracle9*i*AS Personalization includes extensive online help that can be summoned from a list of contents and from Help buttons.

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For additional information, visit the Oracle Accessibility Program web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

JAWS, a Windows screen reader, may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, JAWS may not always read a line of text that consists solely of a bracket or brace.

Conventions

In this manual, Windows refers to the Windows 95, Windows 98, and the Windows NT operating systems.

The SQL interface to Oracle9*i* is referred to as SQL. This interface is the Oracle9*i* implementation of the SQL standard ANSI X3.135-1992, ISO 9075:1992, commonly referred to as the ANSI/ISO SQL standard or SQL92. In examples, an implied carriage return occurs at the end of each line, unless otherwise noted. You must press the Return key at the end of a line of input.

The table below shows the conventions followed in this manual and their meanings:

Convention	Meaning
boldface	Commands, menu names, menu items, names of dialogs.
`code`	Data fields and values, special characters, etc., examples of files, data, filenames, and pathnames.
`italics`	Argument names and placeholders in command formats.
<>	Angle brackets enclose user-supplied names.
`% user input` `system output`	In interactive examples, user input is shown in bold typewriter, and system output is shown in regular typewriter.

1

Introducing Oracle9*i*AS Personalization

This chapter introduces Oracle9*i*AS Personalization, describes what it does and how it does it, and describes its main components.

What Is Oracle9*i*AS Personalization?

Oracle9*i*AS Personalization (OP) is an integrated software application that provides a way for a Web site to customize — or personalize — the recommendations it presents to Web site visitors and customers.

Recommendations are personalized for each visitor to the Web site. This has distinct advantages over tailoring recommendations to broad, general market segments. Recommendations are based on a visitor's data and activity such as navigational behavior, ratings, purchases, as well as demographic data.

OP collects the data and uses it to build predictive models that support personalized recommendations of the form "a person who has clicked links *x* and *y* and who has demographic characteristics *a* and *b* is likely to buy *z*".

OP incorporates visitor activity into its recommendations in real time — during the Web visitor's session. For example, OP records a visitor's navigation through the Web site, noting the links that are clicked, etc. All this is data stored for that visitor. The visitor may respond to a Web site's request to rate something, e.g., a book or a movie; the rating becomes part of the data stored for that visitor. All the Web-based behavior for the visitor is saved to a database, where OP uses it to build predictive models. This data can be updated with data collected in subsequent sessions, thereby increasing the accuracy of predictions.

OP works in conjunction with an existing Web application. The Web application asks OP to record certain activities, and the data is saved by OP into a schema.

The Web application asks OP to produce a list of products likely to be purchased by a Web site visitor; a scored list of recommendations compiled from the visitor's current behavior and from data in another schema is passed to the Web application.

A third schema maintains administrative schedules and activities.

What Kind of Data Does OP Collect?

OP collects four kinds of data:

- navigational behavior
- ratings
- purchases
- demographic data

Of these, navigational behavior allows the most flexibility. It can represent anything the Web application wants to consider a hit (e.g., viewing a page, clicking a link/item, etc.).

Visitors to the Web site are of two types: registered visitors (customers) and unregistered visitors (visitors). For customers, OP has both data from a current session and historical data collected over time for a given customer, as well as demographic data. For visitors, there is no historical data, so recommendations are based on current session behavior and demographic data, if available.

How Does OP Collect the Data?

OP collects the data using Java calls provided by the REAPI (Recommendation Engine Application Programming Interface). These calls add information to the RE cache for the specific session, identified by a session ID. The RE finds the correct session ID by looking up one of the following arguments passed in the REAPI calls:

appSessionID -- used by *sessionful* Web applications (that is, an application that stores an identifier for each session)

customerID -- used by *sessionless* Web applications (that is, an application that does not store an identifier for each session)

In more detail: The data collected are temporarily stored in a dual buffer cache in the JServ (Java server). Periodically the JServ buffer is flushed and the data are sent to the appropriate RE schema. The session data are then used, combined with historical data, to generate recommendations. Finally, the RE instance periodically flushes the data to the MTR for sessions that have concluded or timed out. The RE only flushes data to the MTR with the data source types specified by its

configuration parameters. The data in the MTR is then used to build predictive models for future deployment.

Sessionful and Sessionless Web Applications

Some Web applications are *sessionful*, i.e., they create a session for each user visit to the Web site. Others are *sessionless* (stateless), i.e., they do not create sessions.

Regardless of whether the calling Web application is sessionful or sessionless, OP is always sessionful; OP always creates a session internally and maps that session to the Web site's session if there is one.

During the OP session, the Web application can collect data and/or request recommendations.

What Does OP Do with the Data?

OP uses the data to build data mining models. The models predict what the Web site visitor will probably like or buy. The predictions are based on the data collected for that Web site visitor in previous sessions, in the current session, and on demographic information.

The OP Administrator defines a *package* that contains information needed to build a model or models, as well as information about the database connections. The OP Administrator creates and manages schedules for building the packages, and for deploying the packages to the recommendation engines (REs) that will produce the recommendations. Recommendation engines with the same package are grouped together in recommendation engine farms (RE Farms). These and related terms are defined more fully in the next section.

Models

OP uses one of two algorithms, depending on the type of recommendation requested by the web application. Both algorithms are based on a theorem of Bayes concerning conditional probability. See Appendix A for a description of the algorithms.

OP Components

The user of the OP Administrative UI is assumed to be a DBA or a Java programmer. *Getting Started with Oracle9iAS Personalization* is designed to introduce

a Java programmer or DBA to the basic components and interfaces of OP. These components and interfaces consist of:

- **Administrative UI:** A browser-based user interface that permits the OP Administrator to build, schedule, and deploy packages, manage RE Farms and REs, and otherwise manage OP. Chapter 2 describes the Administrative UI in detail. The Administrative UI is installed on the system where Oracle9*i* is installed.
- **Recommendation Engine (RE):** The RE supports a Web application written in Java for collecting and preprocessing customer and visitor data, and for providing recommendations to those customers and visitors. Access to the RE is provided via the REAPI (Recommendation Engine Application Programming Interface). A given RE may support one or more Java server processes in a Web application.
- **Recommendation Engine Farm (RE Farm):** A logical grouping of one or more REs that are populated with the same deployable package (data mining model(s)). An RE Farm is generally treated as a single unit for management from the Administrative UI.
- **Package:** An object created using the Administrative UI. A package contains the information from historical data necessary to make recommendations. A package defines the build settings and other attributes necessary for building data mining models and to schedule model builds.
- **Mining Object Repository (MOR):** The Oracle database that maintains mining metadata and mining model results as defined by the Oracle9*i*AS Personalization data mining schema. Via the Administrative UI, the MOR serves as the focus for logging in to the data mining system, logging off, and scheduling OP events.
- **Mining Table Repository (MTR):** The MTR contains the schema and data to be used for data mining. The MTR has a fixed schema designed to support the building of models that produce recommendations for customers and visitors.
- **Recommendation Engine API (REAPI):** A collection of Java classes that enable a Web application to collect and preprocess data used to build OP models and to obtain recommendations in real time from OP (that is, to score products for particular customers). OP also includes a Batch RE API that permits users to obtain bulk recommendations in batch, i.e., offline. REAPI and the Batch API are installed on the system where Oracle9*i*AS is installed.

It is an option during OP installation to populate the MTR with a small amount of sample data. If this option is chosen, an RE demo can be accessed and some recommendations and administrative actions can be tested.

Location of OP Components

OP requires both Oracle9*i* and the Oracle9*i* Application Server (Oracle9*i*AS). Oracle9*i* and Oracle9*i*AS are usually installed on different systems.

The following OP components are installed on the system where Oracle9*i*AS is installed:

- REAPI
- RE Batch API
- REAPI Demo

All other components, including the Administrative UI and OP documentation, are installed on the system where Oracle9*i* is installed.

2

The OP Administrative UI

The OP Administrative UI is a browser-based user interface that permits the OP administrator (a Java programmer or DBA) to manage RE Farms and REs, schedule and deploy packages, and otherwise manage OP.

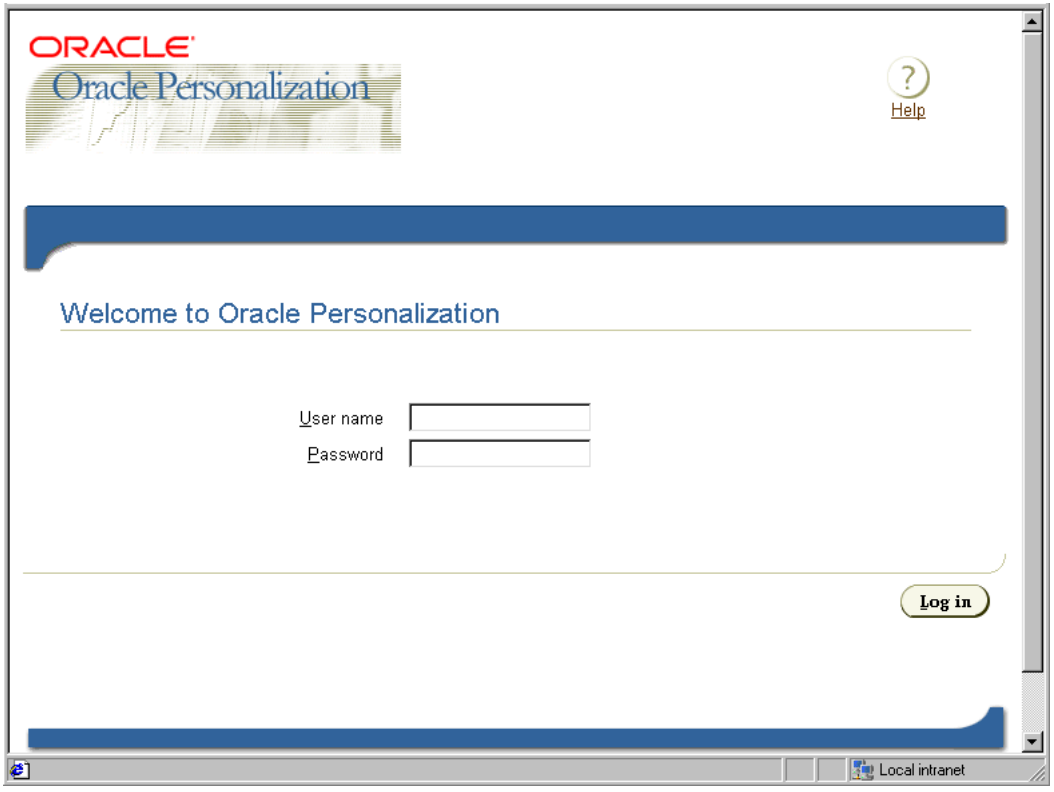
Bring up the OP Administrative UI by typing the following in the URL field of your browser:

```
http://<hostname-oracle9i>/OP/Admin/
```

where `<hostname-oracle9i>` is the name of the system on which Oracle 9*i* and the OP components associated with the database are installed.

Note that the URL is case-sensitive.

The first page that appears is the login page, which welcomes you to Oracle9*i*AS Personalization.



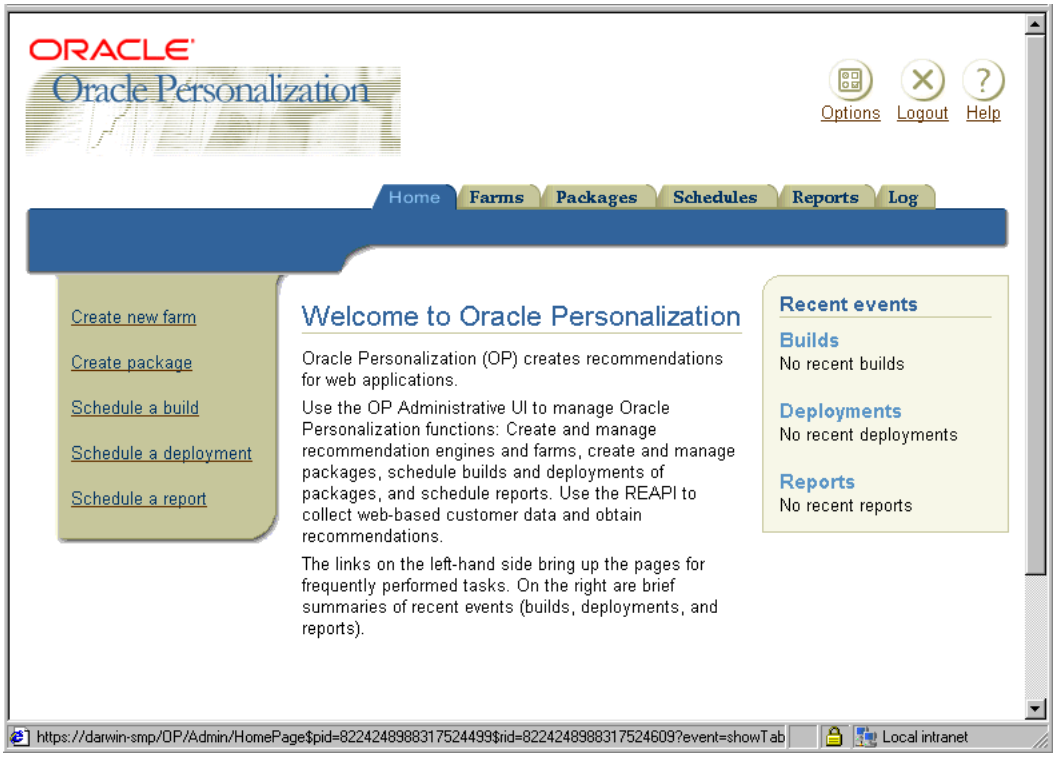
Login Page

Enter user name and password for the MOR schema (a valid user name and password were established during OP installation; if you don't know what they are, ask your DBA).

- After entering a valid user name and password, click Log in, which brings up the OP Administrative home page.

Home Page

The **Home** page welcomes you and briefly describes the product. At the left are links to common tasks, such as create new farm, create package, schedule a build, etc. They are listed in the order you would follow in executing the tasks. To the right is a brief status of recent events (builds, deployments, and reports).



Browse the pages to get familiar with their structure and content. Click the tabs to see what's what and what's where. The tabs are **Home**, **Farms**, **Packages**, **Schedules**, **Reports**, and **Log**. The tab pages organize OP administrative tasks.

Farms Page

The **Farms** page lists the current recommendation engine farms. Use the **Farms** page to create, delete, and in general manage recommendation engine farms.

The **Farms** page gives access to recommendation engines, which you add to a farm. Adding a recommendation engine to a farm means specifying the RE's database connection details. These details were established during the installation of the database.

An RE must have a connection to the MTR. If there is no pre-existing MTR connection, you create one. This also requires information established during installation and configuration of the database.

Packages Page

The **Packages** page lists current packages. Use this page to create, delete, and manage packages.

An OP package contains all the information needed for building predictive models, which includes the general settings for the package, information about its connection to the MTR, and settings specified for the package's build (which includes settings for building the model(s)).

To create a package, click **Create Package**. This brings up the first of three pages that guide you through creating a package:

- The first page, **Create Package**, prompts you for a name and description of the package and asks you to specify the MTR connection. If there is no MTR connection, you must create one. Click Options (upper right), and go to MTR database connections.
- The second page, **Specify Build Settings**, prompts you for the build settings, which include
  - Recommendation performance: This is a trade-off between accuracy and speed.
  - New products: Specify whether you want OP to use a proxy for new products. There must be proxy information in the MTR to use this option.
  - Choose what time-stamped session data to use: Use data from any dates, including data with no dates; a rolling window of the past x number of days, or between specified dates.

- The third page, **Confirm Settings**, summarizes and confirms the settings you have specified.

The **Confirm Settings** page also gives you the option of going directly to scheduling a build for the package.

Schedules Page

Click the **Schedules** tab and note the three kinds of events that are scheduled: Builds, deployments, and reports. The **Schedules** tab opens by default to the **Build Schedule** page.

Build Schedule

Use this page to create, edit, delete, and in general manage the building of a package, that is, the creation of predictive models and other information needed to make recommendations.

To create a new build schedule, click **Create Build Schedule**. The **Create Build Schedule** page gives you the following options:

- Build the package as soon as possible.
- Build the package at a future time that you specify, and with the frequency you specify.

This page also gives you the option of going directly to scheduling deployment of the package.

Deployment Schedule

Use this page to create, edit, delete, and in general manage deployment schedules.

Deploying a package means copying it to every RE on a given RE Farm. Multiple REs with the same package can share the load.To create a new deployment schedule, click **Create Deployment Schedule**. The **Create Deployment Schedule** page gives you the following options:

- Deploy the package every time it is built.
- Deploy the package as soon as possible.
- Deploy the package at a future time that you specify, and with the frequency you specify.

Report Schedule

Use this page to create and manage report schedules.

To schedule a report, click **Create report schedule**.

The **Create Report Schedule** page gives you the following options:

- General settings: Specify the RE Farm, the report type, the email address of the person who should receive notification, and any notes you may wish to make.
- Data selection: Use data from any dates; a rolling window of the past x number of days, or between specified dates.
- Schedule: Run the report as soon as possible or schedule the run for a specified date with specified frequency.

The Create Report Schedule page also lets you do the following:

- To update or look at the details of a report schedule, click its Edit icon.
- To delete a scheduled report, select it and click **Delete**.
- To stop a scheduled report that is running, select it and click **Stop**.

Note: You cannot edit or delete a scheduled report while it is running or canceling; it must be idle.

Reports Page

Use this page to monitor reports. There are three types of reports:

- **Visitor-to-customer conversion:** Reports the number of anonymous visitors who registered and became known customers within a user-specified time period.

  Specifically, it reports the number of unique visitor sessions, the number of unique customer sessions (sessions that had at least one purchase), and the conversion rate (number of customers divided by the number of visitors).
- **Recommendation effectiveness:** Reports, for a specified time period, the number of recommendation requests served by the system, the number of recommended items clicked, the number of recommended items bought, and the percentages of clicked and bought items.
- **Cross-sold items:** Reports, for a specified time period, the number of recommendation requests served by the system, the number of recommended

items clicked, the number of recommended items bought, and the percentage of clicked and bought items.

Note: The MTR tables MTR_SESSION and MTR_RECOMMENDATION_DETAIL must be populated before a report can be generated.

Log Page

The event log allows you to monitor results of scheduled builds, deployments, and reports.

To view details of an item, click its Details icon. To delete one or more items, select the item(s) and click **Delete**.

3

Using the OP Administrative UI

This chapter walks you through the administrative tasks of creating an RE Farm with one recommendation engine, building a package, and deploying that package to the RE Farm.

To perform these steps, you will need information that was established during installation and configuration, such as database and schema names and locations. If the OP Administrative UI isn't still up, bring it up by typing the following in the URL field of your browser.

```
http://<hostname-oracle9i>/OP/Admin/
```

where `<hostname-oracle9i>` is the name of the system on which Oracle 9*i* and the OP components associated with the database are installed.

Log in with a user name and password as advised by your DBA.

Create an RE Farm with an RE

The first step is to create an RE Farm. There are two ways to start:

- On the **Home** page, click the **Create new farm** link to bring up the **Create Recommendation Engine Farm** page.
- Click the **Farms** tab to go to the **Farms** page. On the **Farms** page, click **Create Farm** (lower right), which brings up the **Create Recommendation Engine Farm** page.

On the **Create Recommendation Engine Farm** page,

1. Enter a name for the farm.
2. Click **Add Recommendation Engine**.

3. On the **Add Recommendation Engine** page, enter a name for the recommendation engine.

   For the database connection details, you will need information that was provided during installation.

   - For **Host ID**, **Port**, **SID**, and **Database alias**, you will need the database information you provided when you installed the database.
   - For **DB schema name**, **User name**, and **Password**, you will need the information provided when you installed and configured the RE schema. The default values for these are RE, RE, and REPW.

4. After filling in all the fields, click **Test Connection** to determine whether the database connection is successful.
5. Assuming the database connection is successful, click **OK** (lower right), which takes you back to the **Create Recommendation Engine Farm** page, where you will see listed the recommendation engine you defined.

   Note: If you click **Cancel** instead of **OK**, the information you have entered is lost.

Create a Package

Next, create a package.

To create a package, you must have a connection to the MTR. If you do not have an MTR connection,

1. Click **Options** (upper right) and go to the **MTR database connections** section. The information needed for an MTR connection comes from entries provided when you installed and configured the MTR schema.
2. To check the database connection, click **Test Connection**.
3. Assuming the connection is successful, click **OK**, which returns you to the **Create Farm** page.

Now you can create a package:

1. Click the **Packages** tab, which brings up the **Packages** page.
2. On the **Packages** page, click **Create Package** (lower right), which brings up the first of three pages of the Create Package wizard.
3. Give the package a name (required), a description (optional), confirm the name of the MTR connection, and click **Next**.

4. Specify the settings to be used to build and tune the new package, and click **Next**.
5. Review the settings you have specified, leave the **Schedule a build** box checked, and click **Finish**. This takes you to the **Create Build Schedule** page.

Schedule a Build

On the **Create Build Schedule** page, select

1. **Build as soon as possible**.
2. Leave the **Schedule deployment** box checked.
3. Click **OK**. This takes you to the **Create Deployment Schedule** page.

Schedule a Deployment

On the Create Deployment Schedule page,

1. Select **Deploy every time the package is built**.
2. Leave the default for **Frequency**, Once.
3. Click **OK**. This returns you to the **Packages** page.

Summary

You have created an RE Farm with one RE, and you have created a package and scheduled its build and deployment.

Check back after a few minutes to see whether the package has yet built and deployed. When the package has built and deployed successfully, you can use it to collect data and make recommendations using the Recommendation Engine API.

MTR Sample

Next, browse the contents of the Mining Table Repository (MTR) database used to build the model. This is the prepopulated MTR that is installed when OP is installed if you select that option. This prepopulated MTR provides the data needed to perform the exercises described in this manual.

Use SQL*Plus commands to examine the contents of any of the database tables. The table below shows what part of one of the MTR database tables looks like. It contains movie ratings by customers, demographic data on those customers, an ID for each movie that was rated, the rating given the movie by the customer, and the data source type.

CUSTOMER_ID	ITEM_ID	ITEM_TYPE	ATTRIBUTE_ID	BIN_VALUE	DATA_SOURCE_TYPE
2	264	MOVIE	1	1	2
2	389	MOVIE	2	1	3
2	153	MOVIE	2	2	3
2	354	MOVIE	2	2	3
2	264	MOVIE	2	3	3
2	153	MOVIE	3	1	4
2	264	MOVIE	3	1	4
2	354	MOVIE	3	1	4
2	389	MOVIE	3	1	4
2	0	NONE	4	3	1
2	0	NONE	5	1	1
2	0	NONE	6	2	1

This sample comes from a large database table that contains movie ratings by customers, demographic data on those customers, an ID for each movie that was rated, and the rating given the movie by the customer. Table columns are as follows:

- CUSTOMER_ID:
- ITEM_ID: The ID numbers identify particular movies.
- ITEM_TYPE: Refers to kind of item being rated; here the type is movie.

- ATTRIBUTE_ID: Values 1, 2, and 3 are attributes from purchasing, rating, and navigational data; values 4, 5, and 6 are from demographic data, where 4 = age, 5 = gender, and 6 = marital status.
- BIN_VALUE: Ratings were binned into 3 bins; the values here correspond to those bins.
- DATA_SOURCE_TYPE: (what kind of data) 1 = demographic, 2 = purchases, 3 = ratings, 4 = navigational.

For more information about the OP schemas, see the *Oracle9iAS Personalization Administrator's Guide*, especially Chapter 8.

4

Using the REAPI Demo

This chapter presents several exercises using the REAPI Demo.

Running the REAPI Demo accomplishes the following:

- You can verify that OP was installed correctly by using the REAPI to get recommendations. To run REAPI Demo, OP must be installed properly and the REAPI Demo must be configured to access RE.
- You can experiment with configuration so that you can learn how best to configure RE to suit your Web site.
- You can familiarize yourself with OP's Java operations and see how the calls work and how they interact.
- To follow the exercises in this chapter, you need to have installed a populated MTR (installed if you selected "Install demo data" when you installed OP).
- You can view the source code to see how the program uses the REAPI calls.

For detailed information about the REAPI, see the *Oracle9iAS Personalization Recommendation Engine API Programmer's Guide*.

For details about installing REAPI, see the *Oracle9iAS Personalization Administrator's Guide*.

In real life, the REAPI calls are used to "instrument" a Web site so that you can collect the data needed to create good models and generate recommendations.

REAPI and the REAPI Demo are installed on the system where Oracle9*i*AS is installed.

Before Running REAPI Demo

Before running the REAPI Demo, you need to have created and deployed a package to the recommendation engine so that there is data available for making recommendations.

When you install OP, install the populated MTR.

Start the OP Administrative UI by pointing your browser to the following URL:

```
http://<hostname-oracle9i>/OP/Admin/
```

where `<hostname-oracle9i>` is the name of the system on which Oracle9*i* and the OP components associated with the database are installed.

Next, perform the following steps:

1. Create a movies MTR connection, using the sample movies MTR database supplied with OP.
2. Create a package using the movies MTR connection created in step 1.
3. Create a recommendation engine farm (RE Farm) with one recommendation engine (RE).
4. Build the package.
5. Deploy the movies package to the farm created in step 3.

After the movies package is successfully deployed to the RE Farm, you can begin experimenting with the REAPI demo to see how the REAPI calls work.

Start REAPI Demo

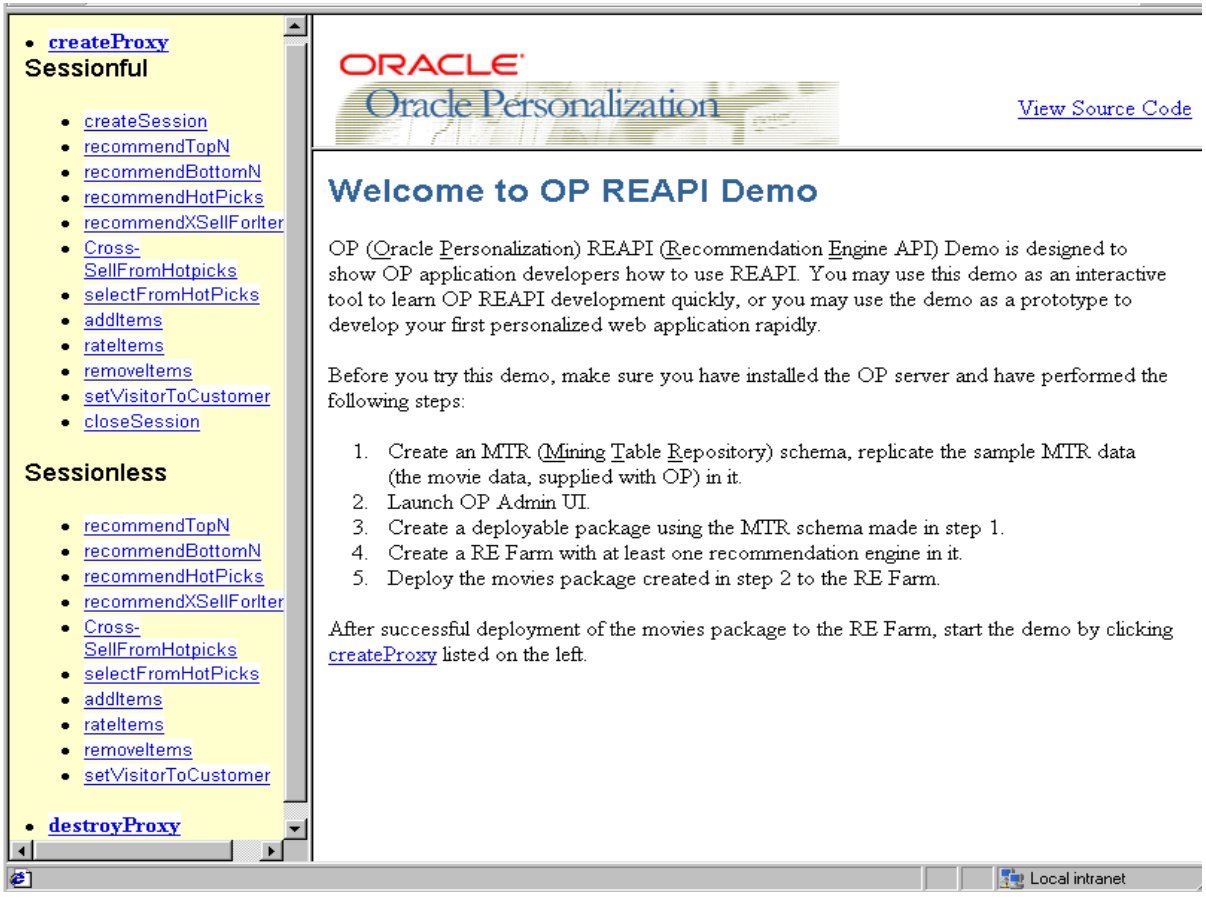
When you are ready to begin working with the REAPI Demo, point your browser to the following URL:

```
http://<hostname-oracle9ias>/redemo/
```

where `<hostname-oracle9ias>` is the name of the system on which Oracle9*i*AS and the OP components associated with the application server are installed. This will bring up the REAPI Demo user interface. The first page that appears is the **Welcome to OP REAPI Demo** page.

Proxies

REAPI calls execute in the context of a proxy, which specifies the environment in which the calls execute. One of the important elements of the environment is the default RE schema. You must create a proxy before you execute any calls.



View Source Code

While you are running the REAPI Demo, you can view the source code for the demo by clicking the **View Source Code** link.

REAPI Demo is implemented as a Java servlet consisting the following classes:

- **REDemoServlet.java**
- **REUtil.java**
- **REDemoException.java**
- **REDemoConstants.java**

You can view the source code for each of these classes; you can also download the classes to your desktop.

Exercises

This section provides exercises that you can work through to get a feel for how the REAPI Demo works. The exercises are as follows:

- Creating a proxy
- A sessionful exercise
- A sessionless exercise
- Changing a visitor to a customer

Values and Results

To get meaningful results from these examples, you must pass "valid" values (that is, values for which there *are* recommendations) to REAPI Demo. All the values in this demo produce useful results. You can determine other values by examining the populated MTR provided with OP.

The results returned when you execute the calls will be similar to but *not* identical to the results displayed in this document.

Exercise 1: Creating a Proxy

You must create a proxy before you execute any calls. Here's how: In the frame on the left of the REAPI Demo screen, click **createProxy**, and enter these values:

- **RE Name**: The name of the recommendation engine schema that you will use.
- **JDBC URL**: `jdbc:oracle:thin:<host>:<port>:<sid>` Enter the values that were specified during OP installation.

- **User name:** The user name for the database. Enter the RE user name that was specified during OP installation.
- **Password:** The password for the RE user. Enter the password that was specified during OP installation.
- **Cache Size:** 3242 KB (default).
- **Archive Interval:** 10,000 milliseconds (default).

Click **Create RE Proxy**.

REAPI Demo displays a message indicating success or failure. (REAPI Demo does this for every action.)

Exercise 2: A Sessionful Web Application

Follow these steps to experiment with any of the methods listed under the heading "Sessionful." (These are the methods that are valid with a sessionful Web application.)

A sessionful Web application starts a session for each customer when the customer logs in to the site.

1. To create a session, specify a customer ID and a session ID. At the left, under **Sessionful**, click **createSession** to bring up the **Create RE Session** page, and enter these values:
   - **User Type**: Customer
   - **User ID:** 1
   - **App Session ID:** 1011 (You must enter a unique number; this number cannot be used more than once.)

   Click **Create Session**.
2. You can now execute any of the methods listed under **Sessionful**, for example, to obtain recommendations. To do this, click **recommendTopN** to bring up the **Recommend Top Items** page, and enter these values:
   - **Number of recommendations to display:** 10
   - **Tuning Settings**
     - **Data Source Type:** PURCHASING
     - **Interest Dimension:** PURCHASING
     - **Personalization Index Type:** LOW

  - **Profile Data Balance**: CURRENT
  - **Profile Usage**: INCLUDE
- **Filtering Settings**
  - **Filtering Type**: All Items
  - **Taxonomy ID**: 1
  - **Select Categories:** For Include items, you can highlight any individual items, or, if you set Filtering Type to All Items, you do not need to highlight any items or categories.
- **Recommendation Content**
  - **Sorting order for Prediction:** ASCEND
  - **Sorting order for Type**: NONE
  - **Sorting order for ID**: NONE

  Click **Recommend Top N**.

OP then returns 10 or fewer recommendations for the user of this session. It will look something like the table below, which displays information for four movies. It displays the item type (MOVIE, for each), the item ID (a unique number for each item), and the ranking of the four movies, ranked on the likelihood of being purchased):

Recommended Top N Items for User ID = 1

Type	Item ID	Rating
MOVIE	345	1.0000
MOVIE	383	2.0000
MOVIE	447	3.0000
MOVIE	223	4.0000

3. You can collect data while a session is running. To add data, click **addItems** to bring up the **Add Items** page, and enter these values
   - **Data source Type**: RATING
   - **Item Type**: MOVIE
   - **Item ID**: 122

- **Value:** 5 (Rating on a scale from 1 to 5.)

Click **Add>>** to move this item to the list; click **Add Items** to add it to the RE. This updates the information in the RE tables for this user.

4. Next, try Hot Picks. Hot picks are used by some Web sites; for example, daily specials might be in a Hot Picks Group. Click **recommendHotPicks** to bring up the **Recommend from Hot Picks** page, and enter these values:

   - **# of Recommended Items:** 10
   - **Hot Pick Group**: COMEDY
   - **Tuning Settings**
     - **Data Source Type**: NAVIGATION
     - **Interest Dimension**: NAVIGATION
     - **Personalization Index Type:** LOW
     - **Profile Data Balance**: HISTORY
     - **Profile Usage:** INCLUDE
   - **Filtering Settings**
     - **Filtering Type:** Exclude Items
     - **Taxonomy ID**: 1
     - **Select Categories:** For Exclude items, you do not need to select here.
   - **Recommendation Content**
     - **Sorting order for Prediction**: ASCEND
     - **Sorting order for Type:** NONE
     - **Sorting order for ID**: NONE

   Click **Recommend Hot Picks**.

   OP then returns 10 or fewer recommendations for the user of this session. In this case, it returns something like what is shown in the table below. Again, it

displays the item type (MOVIE), the item ID (a number identifying each movie), and the rating.

Recommended Hotpick Items for User ID = 1

Type	Item ID	Rating
MOVIE	360	1.0000
MOVIE	370	2.0000
MOVIE	116	3.0000
MOVIE	83	4.0000
MOVIE	28	5.0000
MOVIE	20	6.0000

5. You can also rate items with respect to the current user, that is, determine how the current user will rate items. Click **rateItems** to bring up the **Rate Items** page, and enter these values:
   - **Taxonomy ID**: 1
   - **Tuning Settings**
     - **Data Source Type**: RATING
     - **Interest Dimension**: RATING
     - **Personalization Index Type**: MEDIUM
     - **Profile Data Balance**: HISTORY
     - **Profile Usage**: INCLUDE
   - **Recommendation Content**
     - **Sorting order for Prediction**: ASCEND
     - **Sorting order for Type**: NONE
     - **Sorting order for ID**: NONE
   - **Items**
     - **Item Type**: MOVIE
     - **Item ID**: 72

Click **Add>>** to add this item to the list. You can add other items to the list, if you wish.

- Similarly, add "Movie, 122" and "Movie, 287".
- When you are finished adding items, click **Rate Items**.

REAPI Demo returns the following:

6. Rate Items for User ID = 1

Type	Item ID	Rating
MOVIE	72	4.3071
MOVIE	287	4.3453
MOVIE	122	4.3569

7. When you have finished experimenting, close the session. (REAPI sessions that are not explicitly closed eventually time out; it is a good practice, however, to close them explicitly.) Close the session by clicking **closeSession**, which brings up the **Close the Current Session** page. Click the **Close Session** button.

Exercise 3: A Sessionless Web Application

If your Web site does not start a session for each visitor or customer, you use the calls listed under **Sessionless**. For each of these calls, you provide the identification data for user and session. Otherwise, the calls are identical to sessionful ones. This example illustrates several sessionless calls.

1. If a customer buys an item, you may want to offer the customer a similar or related item, that is, a cross-sell item. To create cross-sell recommendations for an item from the HORROR hot picks group, click **Cross-SellFromHotpicks** in the left frame under **Sessionless**. The **Cross-Sell for Items from Hot Picks** page is displayed.
   - **Number of recommendations to display:** 10
     - **Hot Pick Group:** HORROR
   - **User Details**
     - **User Type:** Customer
     - **User ID:** 1
   - **Tuning Settings**

  - **Data Source Type:** NAVIGATION
  - **Interest Dimension:** NAVIGATION
  - **Personalization Index Type:** MEDIUM
  - **Profile Data Balance:** CURRENT
  - **Profile Usage:** INCLUDE
- **Filtering Settings**
  - **Filtering Type:** Exclude Items
  - **Taxonomy ID:** 1
  - **Select Categories:** Select items you want to exclude or select none.
- **Recommendation Content**
  - **Sorting order for Prediction:** ASCEND
  - **Sorting order for Type:** NONE
  - **Sorting order for ID:** NONE
- **Items**
  - **Item Type:** MOVIE
  - **Item ID:** 199

    Click **Add>>** to add this item to the list. Similarly, add MOVIE 354 and MOVIE 122 to the list.

When you have finished adding items, click **Cross-Sell for Items from Hot Picks**.

OP then returns 10 or fewer recommendations for the user of this session. In this case, it returns the following:

Cross-Sell for Items for Hot Picks for User ID = 1

Type	Item ID	Rating
MOVIE	72	1.2785

Exercise 4: Change Visitor to Customer

If a visitor registers as a user during a session, you need to change the visitor to a customer. You can only change a visitor to a customer during a session that the user entered as a visitor.

1. To create a session using the default proxy, you specify a visitor ID and a session ID. At the left, under **Sessionful**, click **createSession** to bring up the **Create RE Session** page, and enter these values:
   - **User Type:** Visitor
   - **User ID:** 100
   - **App Session ID:** 1015

     Note that **App Session ID** value must refer to a session that does not already exist.
2. Now you can change the visitor to a customer. At the left, under **Sessionful**, click **setVisitorToCustomer** to bring up the **Set Visitor to Customer** page. Click **Set Visitor to Customer**.

   REAPI Demo displays a message that announces SUCCESS!

You can close the session now or continue experimenting, as you like.

Summary

This chapter has presented exercises to show you how the REAPI works.

The Demo is different from the way you would use the REAPI calls in practice. In practice, you would embed the REAPI calls in a Java program that you write, and you would execute the program as you ordinarily do.

A

Recommendation Algorithms

This appendix contains descriptions of the two recommendation algorithms used by Oracle9*i*AS Personalization to create models. Models are used to generate personalized recommendations. The two algorithms are

- Predictive Association Rules
- Transactional Naive Bayes

Predictive Association Rules

The most familiar use of association rules is what we know as "market basket analysis," i.e., rules about what goes with what in a shopping cart, such as "eighty percent of people who buy beer also buy potato chips."

The association rules algorithm finds combinations of items that appear frequently in transactions and describes them as rules of the following "if-then" form:
"If A, then B."
where A is the antecedent and B is the consequent. (Note that the two sides of the proposition can be more than one item each; for example, "If A, B, and C, then D and E." For Predictive Association Rules, there is only one item in the consequent.)

It turns out that many such rules can be found -- the challenge is to find those that are meaningful or interesting and that also lead to actionable business decisions. An example is "eighty percent of people who buy beer and pretzels also buy chocolate." This combination is not obvious, and it can lead to a change in display layout, e.g., moving the chocolate display closer to where beer is on sale.

On the other hand, a rule like "eighty percent of people who buy paint also buy paint brushes" is not very useful, given that it's obvious and doesn't lead you to change the arrangement of these items in your store -- they're probably already displayed near each other.

Similarly, "eighty percent of people who buy toothpaste and tissues also buy tomatoes" is not obvious, and is probably not useful as it may not lead to any actionable business decision.

To identify rules that are useful or interesting, three measures are introduced: support, confidence, and lift.

Support: First, determine which rules have strong support, i.e., rules that are based on many examples in the database. Support is the percentage of records that obey the rule, i.e., baskets that contain both A and B.

Confidence: Next, determine which rules have high confidence, i.e., instances that obey the rule (contain both A and B) as a percentage of all instances of A. For example, assume you have 10 instances of A, 8 of which also have B; the other 2 do not have B. Confidence is 8 out of 10, or 80 percent.

Lift: Lift compares the chances of having B, given A, to the chances of having B in any random basket. Of the three, lift is the most useful because it improves predictability.

Transactional Naive Bayes

Naive Bayes is a type of supervised-learning module that contains examples of the input-target mapping the model tries to learn. Such models make predictions about new data based on the examination of previous data. Different types of models have different internal approaches to learning from previous data. The Naive Bayes algorithm uses the mathematics of Bayes' Theorem to make its predictions.

Bayes' Theorem is about conditional probabilities. It states that the probability of a particular predicted event, given the evidence in this instance, is computed from three other numbers: the probability of that prediction in similar situations in general, ignoring the specific evidence (this is called the *prior* probability); times the probability of seeing the evidence we have here, given that the particular prediction is correct; divided by the sum, for each possible prediction (including the present one), of a similar product for that prediction (i.e., the probability of that prediction in general, times the probability of seeing the current evidence given that possible prediction).

A simplifying assumption (the "naive" part) is that the probability of the combined pieces of evidence, given this prediction, is simply the product of the probabilities of the individual pieces of evidence, given this prediction. The assumption is true when the pieces of evidence work independently of one another, without mutual interference. In other cases, the assumption merely approximates the true value. In practice, the approximation usually does not degrade the model's predictive

accuracy much, and it makes the difference between a computationally feasible algorithm and an intractable one.

Compared to other supervised-learning modules, Naive Bayes has the advantages of simplicity and speed. It also lends itself to future extensions supporting incremental learning and distributed learning.

"Transactional Naive Bayes" refers to the way the input is formatted; the algorithm is the same. The table below shows an example of traditional data format, with columns for the items (customer, apples, oranges, pears, and bananas) and rows for the customers (Joe, Jim, Jeff), and zeroes or ones in each table cell, indicating whether, for example, Joe bought an apple (no), an orange (no), a pear (no), or a banana (yes):

	apples	oranges	pears	bananas
Joe	0	0	0	1
Jim	1	0	0	1
Jeff	0	1	0	0

Traditional data layout often produces a sparse matrix because of all those zeroes; it takes up more space in the database, and therefore takes more time in calculations.

Transaction-based format has basically two columns: customer and "hits." For Joe, the table cell contains "bananas":

Joe	bananas
Jim	apples, bananas
Jeff	oranges

Transactional format looks like a "shopping basket" rather than a checklist and is better in cases where the customers buy only subsets of products. Transactional format has the advantage of being the way the data is stored in the database for this type of problem.

Glossary

Admin UI (Administrative UI)

A graphical user interface that enables you to manage Oracle9*i*AS Personalization, which includes (1) scheduling the build and deployment of packages and the generation of reports, and (2) managing the creation and use of recommendation engines (RE) and RE Farms.

Aggregated Model

This type of model is used for all the recommendation methods except cross-sell. It also allows all types of data source as inputs for predicting any of the interest dimensions. See also Cross-Sell Model.

Algorithm

See Recommendation Algorithms.

Category

An element in a taxonomy; an abstraction for a group of items or categories. In OP, any item or category can belong to one or more other categories. See also Taxonomy.

Category Membership

Category membership specifies how items and categories are related to other categories. For example, an item can have a SUBTREELEAF relation to a category if it is a descendant of that category. Similarly, a category can have a SUBTREENODE or LEVEL relationship with another category. See also Taxonomy.

Cross-Sell Model

This type of model is used only in the cross-sell methods. It allows only either navigational or purchasing types of data source for input, and requires that the interest dimension be directly related to the type of input data source. See also Aggregated Model.

Data Source Types

OP uses data from four types of sources: ratings, purchasing, navigational, and demographic.

Demographics

The general characteristics of a human population and population segments, such as size, growth, density, distribution, and vital statistics, especially when used to identify consumer markets. This data is usually obtained from a customer database.

The particular demographic attributes of interest to OP are listed below. They are stored in the MTR in the CUSTOMER's table/view, which consists of the following fields. Note that customers can be companies as well as individuals.

CUSTOMER_ID

NAME

GENDER

AGE

MARITAL_STATUS

PERSONAL_INCOME

HEAD_OF_HOUSEHOLD_FLAG

HOUSEHOLD_INCOME

HOUSE_HOLD SIZE

RENT_OWN_INDICATOR

ATTRIBUTE1 - ATTRIBUTE50: These are generic attributes that can be mapped to any column in the customers' database or can be null. They provide extra flexibility. The first 25 are for string (VARCHAR2) data; the last 25 (26-50) are for numeric data.

Deployment

The process of transferring the tables that define models to one or more recommendation engines after the models have been built. A deployment also establishes the necessary connections between the recommendation engine and the MTR.

Farm

See Recommendation Engine Farm (RE Farm).

Hot Picks

On some e-commerce sites, vendors promote certain products called "hot picks"; the hot picks might, for example, be this week's specials. The hot pick items are grouped into *hot pick groups*. Hot picks and hot pick groups are specified and maintained by the OP administrator.

I-I

The term I-I is encountered in some detailed error messages. It stands for Item-to-Item, and is an obsolete term for what is now discussed under cross-selling. See Cross-Sell Model.

Interest Dimension

Specifies the interest dimension that items should be ranked against. The interest dimensions supported in OP are rating, purchasing, and navigation.

Mining Object Repository (MOR)

The MOR is the Oracle database that maintains mining metadata defined by the Oracle9*i*AS Personalization data mining schema and serves as the focus for logging in to the data mining system, logging off, and validating users for the MOR and data mining functionality. Provides core data mining algorithm functionality.

Mining Table Repository (MTR)

The MTR contains the schema and data used for data mining. For OP, the MTR has a fixed schema designed to support the building of models that support producing customer / visitor recommendations. The MTR also stores customer data collected through the REAPI.

Model

See Recommendation Algorithms.

OP

Oracle9*i*AS Personalization.

Package

An object created using the Admin UI that controls model building and deployment. A package specifies the build settings and other attributes that control how models are to be built, as well as the RE Farm to which the models are to be deployed. After the build is complete, it consists of the rules tables that are deployed to the recommendation engine.

Personalization Index

The relative degree of individualization desired in OP's recommendations. A high setting produces the most individualized recommendations, those most highly related to the given user profile. A low setting generates recommendations that are the most popular or common for a given user profile. A low setting would yield "best seller" kind of recommendations, whereas a high setting will produce recommendations that may not be appropriate for many people, but the recommendations may be of higher perceived value.

P-I

The term P-I is encountered in some detailed error messages. It stands for Person-to-Item, and is an obsolete term for what is now discussed under aggregated models. See Aggregated Model.

Profile

The collection of data elements available about a user. Profiles are stored in the MTR or cached in the RE.

Rating scale

The rating scale for OP should be in ascending order of "goodness". That is, create a scale in which a high rated item indicates that the user prefers that item over items with lower ratings.

Recommendation Algorithms

Oracle9*i*AS Personalization bases its recommendations on one of two algorithms: Predictive Association Rules and Transactional Naive Bayes:

Predictive Association Rules:

- Models are based on Association Rules

- Builds models and scores datasets inside the database
- The mined rules are stored in database tables
- The rule consequents are combined and ordered using a ranking function
- Consequents with higher ranks are returned as recommendations

Transactional Naive Bayes:

- Models are based on conditional probabilities
- Builds models and scores datasets inside the database
- Probabilities are combined according to Bayes' Theorem and a score is computed
- Products with higher scores are returned as recommendations

For fuller descriptions of these algorithms, see Predictive Association Rules and Transactional Naive Bayes, in Appendix A.

Recommendation Engine (RE)

The front end of Oracle9*i*AS Personalization. Via the REAPI (Recommendation Engine Application Programming Interface), RE provides the following services on a Web server associated with the calling Web application:

- Collects and pre-processes users' profile data
- Returns personalized recommendations

Recommendation Engine API (REAPI)

A collection of classes that enable a Web application written in Java to collect and preprocess data used to build OP models and to obtain recommendations from OP (that is, to Score OP models).

Recommendation Engine Farm (RE Farm)

A group of systems with related OP recommendation engines installed. When a package is deployed to an RE Farm, it is deployed to all members of the RE Farm. See also Web Farm.

Recommendations

This is how Oracle9*i*AS Personalization makes its recommendations:

- Uses demographics, taxonomies, session data histories, current session data, keywords or concepts derived from the items

- Data does not have to be complete
- Performs efficient and scalable scoring
- Builds models offline
- Handles both registered customers and visitors
- Automatically selects algorithm (Transactional Naive Bayes or Predictive Association Rules) for a given recommendation task and context

RE Farm

See Recommendation Engine Farm (RE Farm).

Schedule Item

An object created using the Admin UI that controls when models specified by a package are to be built or deployed, or when a report is to be generated.

Score

With reference to applying a predictive model to new data, scoring means assigning a score that reflects the likelihood that a particular record belongs in a certain class.

Session

Sessions are used to organize user activities. A session corresponds to a set of activities that a user does in "one sitting". Each session is uniquely associated with a user and has a start_time and end_time. All the activities performed by that particular user within the (start_time, end_time) interval are considered to be part of that session.

Sessionful and Sessionless Applications

These terms apply to the host Web application and indicate whether the application does its own session management or not. OP has its own internal session management in both cases. If the application is sessionful, OP maintains the mapping between its internal session and the application's session. If the application is sessionless, OP's session starts at the first activity of the user and ends when the user has been inactive for a pre-specified time period. See also Web Application Session versus OP Session.

SID

See System Identifier.

Status (of recommendation engines)

Recommendation Engines can be in any of the following states:

- **Unloaded:** An RE is first created and no package is deployed to it
- **Loaded:** A package is being deployed to the RE
- **Online:** The RE is ready to provide recommendations
- **Offline:** The RE is not ready to provide recommendations
- **Updating:** Online, but in the process of deploying a package to the RE
- **Switching:** Offline; changing status

System Identifier

An identifier for an Oracle database instance. In OP, it refers to the unique identifier assigned to each system associated with an MOR. Each system attached to an MOR must have a unique identifier specified in its configuration file.

Taxonomy

In the OP context, this term refers to the structural organization of items in a company's catalog or site. Typically the catalog and / or the site has a hierarchical structure like a tree or collection of trees (branching from broader groups at the top all the way down to individual items as the leaves). Items can belong to more than one category and to different levels of the structure. The structure of the OP taxonomy is a DAG (direct acyclic graph), which can contain multiple top-level nodes. The different portions of the taxonomy can be disconnected too. Any node can connect to any other node but there cannot be a path that connects a node's child back to the node itself. OP also supports multiple taxonomies (different ways of organizing the items). The taxonomy is implemented using a group of tables (they are all specified by the customer at installation time):

- MTR-TAXONOMY: Lists the different taxonomies used by the customer.
- MTR-TAXONOMY_CATEGORY: Specifies which categories belong to the different taxonomies. (A category can belong to multiple taxonomies; however, for a given taxonomy, there can be only one instance of any category.)
- MTR-TAXONOMY_CATEGORY_ITEM: Specifies which items go with a given taxonomy, category pair.
- MTR-CATEGORY: Specifies the different categories used by the customer.

User (of OP)

The user of Oracle9*i*AS Personalization is a DBA or system administrator or Java programmer, or perhaps all three. Do not confuse this with the user of a Web site that uses OP.

User (of Web site)

Anyone who visits or logs on to the Web site. There are two kinds of users:

- Customers — registered users of the Web site (typically, the Web site stores information about these users (purchasing history, likes and dislikes, etc.)
- Visitors — users of the Web site who are *not* registered (typically the Web site stores less information about visitors than it does about customers). In many instances, the only information for visitors is the immediate Web navigation. OP synchs data for visitors, but doesn't load it back for scoring.

Either type of user is assigned a user ID by the Web application.

Sometimes the Web site user is referred to as the *end user*, to distinguish it from the user of OP.

Web Application Session versus OP Session

Some Web applications keep track of sessions, which provide an association between the Web server and a Web client. This association persists over multiple connections and/or requests during a given time period.

Sessions are used to maintain state and user identity across multiple page requests. The Web applications maintain session information in different ways, e.g., by using cookies, by URL rewriting, or via hidden variables like HttpSession objects. A Web application is sessionful if it keeps track of sessions, sessionless if it does not.

OP has its own session management. An OP session maps the OP end user's activities during a certain period, i.e., from the first activity until the session is timed out or is closed by the host application. Whether the host Web application is sessionful or sessionless, OP always manages its own session in order to provide better predictions. If the host application is sessionful, the OP session is perfectly mapped to the host application session. If the host application is sessionless, OP tracks the session on its own, which has no effect on the host application.

Web Farm

A Web farm uses two or more servers to host the same site. HTTP requests are usually routed to each server using some appropriate scheme, such as round-robin

routing, to distribute load and allow the site to handle more requests in a timely manner.

During a given session, recommendation requests go to a recommendation engine, because information is temporarily stored there before being synchronized back to the MTR.