Oracle® Application Server Personalization

User's Guide

10*g* Release 2 (10.1.2)

B14052-01

July 2005

ORACLE®

Oracle Application Server Personalization User’s Guide, 10*g* Release 2 (10.1.2)

B14052-01

Contents

Preface

Oracle Application Server Personalization provides real-time personalization for Web sites using an integrated real-time recommendation engine that is embedded in an Oracle database. OracleAS Personalization 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 OracleAS Personalization.

Intended Audience

This manual is intended for users of OracleAS Personalization:

- Database administrator (DBA) who administers OracleAS Personalization
- Application designer/architect
- Application programmer

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible, with good usability, to the disabled community. 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. Accessibility 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 more information, visit the Oracle Accessibility Program Web site at

`http://www.oracle.com/accessibility/`

Accessibility of Code Examples in Documentation

Screen readers 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, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

TTY Access to Oracle Support Services

Oracle provides dedicated Text Telephone (TTY) access to Oracle Support Services within the United States of America 24 hours a day, seven days a week. For TTY support, call 800.446.2398.

Requirements

To follow the exercises in this manual, you will need to have OracleAS Personalization installed, including the option of installing the REAPI demo. (During installation, you are asked whether you want to install the demo data; you do.)

OracleAS Personalization needs both the Oracle database and Oracle Application Server installed, possibly on different systems. If they are installed on different systems,

- install the MOR, MTR, and RE schemas on the machine where the Oracle database is loaded
- install REAPI and the REAPI demo on the machine where Oracle Application Server is loaded
- the Administrative UI operates from the OracleJServ/HTTP server

The following software is required for the Administrative UI browser:

- Netscape 4.75 or Internet Explorer 5.5 with SP1.

Where to Find More Information

Documentation for OracleAS Personalization at the current release consists of the following documents:

- *Oracle Application Server Release Notes,* 10*g* Release 2 (10.1.2), which contains a chapter for each component of Oracle Application Server. The chapter for the OracleAS Personalization component contains platform-specific information, a bug report, and information about any late-breaking changes.
- *Oracle Application Server Personalization User's Guide,* 10*g* Release 2 (10.1.2) (this document).
- *Oracle Application Server Personalization Administrator's Guide,* 10*g* Release 2 (10.1.2).
- *Oracle Application Server Personalization Programmer's Guide,* 10*g* Release 2 (10.1.2). A programmer's manual for programming the recommendation engines in real time and for obtaining bulk recommendations.
- The API classes and methods are also described in Javadoc (Oracle Application Server Personalization Java API Reference), updated for the current release.

Related Manuals

OracleAS Personalization documentation is a component of the Oracle Application Server 10*g* Release 2 (10.1.2) Documentation Library. See especially:

- *Oracle Application Server Concepts*

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

Documentation Formats

Documentation for OracleAS Personalization is provided in PDF and HTML formats.

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

The OracleAS Personalization Administrative UI includes extensive online help that can be summoned from a list of contents and from Help buttons.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Introducing OracleAS Personalization

Oracle Application Server Personalization (OracleAS Personalization) enables 1:1 marketing for businesses by dynamically serving personalized recommendations in real time to both registered customers and anonymous visitors. OracleAS Personalization uses data mining technology to analyze the large amounts of data gathered from Web sites and other applications to find patterns within purchase, demographic, ratings, and navigational data. OracleAS Personalization answers such questions as "Which items is this person most likely to buy or like, and with what likelihood?", "People that bought this item are likely to buy which other item?", and "What should I offer this customer to retain his or her business?"

This chapter first explains what personalization is and then introduces OracleAS Personalization.

1.1 What Is Personalization?

Personalization provides relevant and specific recommendations for individuals, taking into account personal preferences, demographics, and behavior. Personalization permits delivering recommendations with the touch and timing of someone who knows you well.

1.2 What Is OracleAS Personalization?

OracleAS Personalization is an integrated software product that provides a way for businesses to derive or generate recommendations for their customers.

OracleAS Personalization makes recommendations using data mining technology without the need to explicitly define manual rules. It should not be confused with similar processes such as customization, business rules, or memory-based collaborative filtering.

Customization requires users to explicitly state preferences such as which stocks or sports teams to track. Personalization automatically deduces the customer’s interests from the customer’s behavior.

Business rules, such as "people who buy digital cameras buy many batteries for the cameras," are created from the experience of human beings running a business. They are not automatic and do not normally apply to a particular customer.

Memory-based collaborative filtering considers a customer’s site usage history in locating like-minded customers for purchases. The entire dataset or summaries are searched which can adversely impact performance.

OracleAS Personalization recommendations are personalized for each customer. A customer can be an anonymous visitor or registered customer at a Web site, or can also be a customer calling in to a call center, using an ATM machine, or using a wireless phone. To serve recommendations, applications make Java-API calls to OracleAS Personalization.

OracleAS Personalization collects customer profile data and uses them to build predictive models that support personalized recommendations. The underlying rules derived through data mining can be more sophisticated and thus yield better results than the other techniques noted above. For example, "a person who has clicked links *x* and *y* and who has demographic characteristics *a* and *b* is likely to buy *z*."

OracleAS Personalization incorporates customer activity into its recommendations in real time, for example, during a customer's Web session. Using OracleAS Personalization, the application records a customer's navigation through the Web site noting the links that are clicked, customer responses to requests to rate items such as books or movies, customer purchases, and any updates to customer demographic information. All this data becomes part of the customer's profile which is stored in the Oracle database. OracleAS Personalization uses all customers' profiles to build predictive models. This data can be updated with data collected in subsequent sessions, thereby increasing the accuracy of predictions when new models are built.

OracleAS Personalization can work in conjunction with an existing Web application or other customer applications. The application asks OracleAS Personalization to produce a list of items likely to be of interest to the customer. A scored list of recommendations is returned to the application.

Although recommendations to Web site users is one important use of OracleAS Personalization, OracleAS Personalization can provide recommendations in other situations. Any application that collects customer data and needs to provide personalized recommendations can use OracleAS Personalization.

In addition to the real-time recommendations interface, OracleAS Personalization also has a batch interface that can be used to generate recommendations offline. Such recommendations would be useful in, for example, marketing campaigns.

1.3 OracleAS Personalization Components

The OracleAS Personalization components consist of:

- **Recommendation Engine Application Programming Interface (REAPI):** A collection of Java classes that enables an application to provide profile data used to build OracleAS Personalization predictive models and to obtain recommendations in real-time from OracleAS Personalization from Recommendation Engines.
- **Recommendation Engine (RE):** An RE consists of programs and tables (RE schema) in an Oracle database. The RE supports an 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 through the REAPI. 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 predictive model package. An RE Farm is generally treated as a single unit for management from the Administrative UI. Also, the REs in the RE Farm can be in separate databases for improvements in scalability and availability.
- **Administrative UI:** A browser-based user interface that permits the OracleAS Personalization Administrator to schedule package builds, deployments, and

reports, manage RE Farms and REs, and otherwise manage OracleAS Personalization. Chapter 2 describes the Administrative UI in detail.

- **Mining Object Repository (MOR):** The MOR holds all the programs and tables that are responsible for OracleAS Personalization's administrative functions. Access to the MOR is provided by the Administrative UI. MOR is the schema that maintains the predictive model packages, package settings, and build and deployment schedules for packages. It contains the algorithms that build the predictive models.
- **Predictive Model Package:** An object created using the Administrative UI. A predictive model package contains data mining predictive models derived from historical customer profile data. The administrator defines the package, which is scheduled for deployment through the Administrative UI. After the package build is complete, the package consists of the predictive model tables that are deployed to the recommendation engines in the specified RE Farm (see Figure 1–1). This is also referred to as just "package" in this document.
- **Mining Table Repository (MTR):** The MTR contains the schema and all data to be used for data mining. The MTR has a fixed schema designed to support the building of predictive models.
- **Batch Recommendation Engine API**: A collection of Java classes that permits users to obtain recommendations in batch, that is, offline.

1.3.1 Location of OracleAS Personalization Components

OracleAS Personalization requires both Oracle Database and Oracle Application Server. Oracle Databases and Oracle Application Server can either be installed on the same system or on different systems.

The Administrative UI runs on the same system as Oracle Application Server.

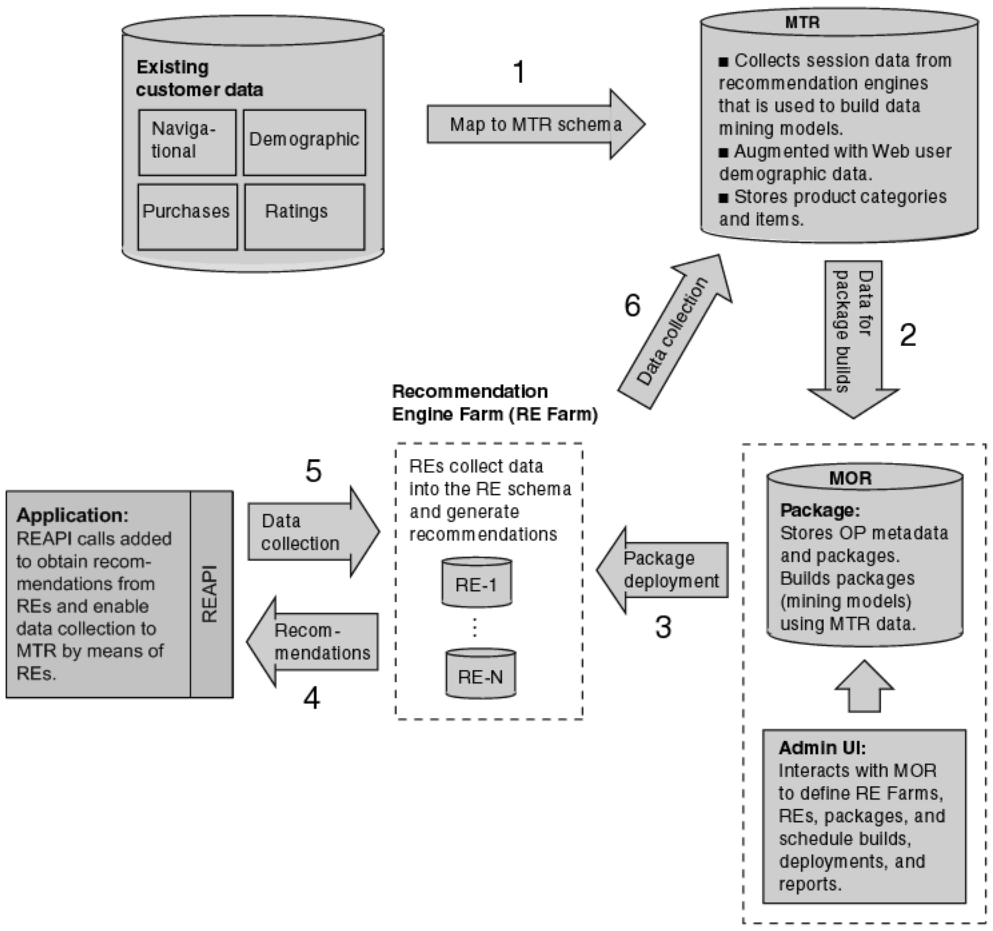
The REAPI and RE Batch API exist on the same system as the applications using them. These applications need not necessarily run on same system as Oracle Application Server.

All other components are installed on the system where the database is installed. The MTR, MOR, and RE need not be installed in the same database. For improved scalability and availability, they may be installed on separate Oracle databases.

1.4 How It All Works

OracleAS Personalization's components and process are depicted in Figure 1–1. The diagram is a flow chart of the entire OracleAS Personalization process.

Figure 1–1 OracleAS Personalization Process


Existing customer data
Naviga-tional
Demographic
Purchases
Ratings
1
Map to MTR schema
MTR
■ Collects session data from recommendation engines that is used to build data mining models.
■ Augmented with Web user demographic data.
■ Stores product categories and items.
6
Data collection
Data for package builds
2
Recommendation Engine Farm (RE Farm)
REs collect data into the RE schema and generate recommendations
RE-1
RE-N
5
Data collection
Application:
REAPI calls added to obtain recom-mendations from REs and enable data collection to MTR by means of REs.
REAPI
Recom-mendations
4
Package deployment
3
MOR
Package:
Stores OP metadata and packages. Builds packages (mining models) using MTR data.
Admin UI:
Interacts with MOR to define RE Farms, REs, packages, and schedule builds, deployments, and reports.

Existing customer data is mapped to the MTR schema (step 1). The data in the MTR is used to build a package in the MOR (step 2). Once that package is built, it is deployed to the recommendation engine farm (step 3), where it is used to generate recommendations that are then passed to the application (step 4). The application collects data through the REAPI data collection methods, which are passed back to the REs (step 5) and are eventually synchronized back to the MTR (step 6).

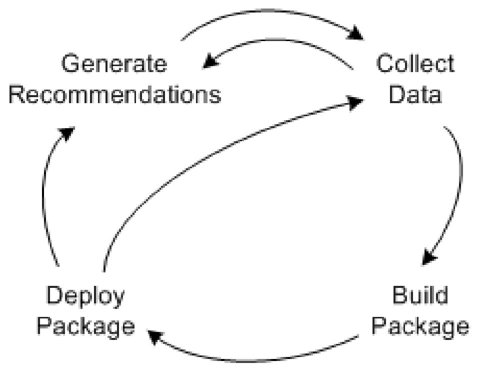
Populating the MTR

There are two ways of getting data into the MTR. The first way is to map existing customer profile data (navigation, purchasing, ratings, and demographics) from application databases onto the MTR schema using database views. This method enables you to build packages immediately and start generating recommendations. However, if you do not have existing profile data, you can use the seed data that is optionally installed with OracleAS Personalization to initialize the MTR. A predictive model cannot be built without data in the MTR.

The second way to get data into the MTR is through the REAPI data collection methods, which are called by an application. As a customer or visitor interacts with your application, your application makes calls to the REAPI to collect data in the RE. The RE then synchronizes the data back to the MTR. Subsequent package builds are made using this data.

Note that to collect data using REAPI calls, there must be a deployed package in the RE. Once you have built and deployed a package, you can collect data using the REAPI. Figure 1–2 provides a graphical depiction of the process cycle.

Figure 1–2 Data collection, package build and deploy, generate recommendations cycle



1.5 What Kind of Data Does OracleAS Personalization Collect?

OracleAS Personalization collects the following kinds of data:

- navigation data
- ratings data
- purchasing data
- demographic data

Of these, navigational data allows the most flexibility. It can represent anything an application wants to consider that is of interest to the customer. For example, in a Web application, a "hit" may be a customer viewing a page or clicking a link/item.

Navigation, ratings, purchasing, and demographics data are also known collectively as *customer profile data*.

1.6 How Does OracleAS Personalization Collect the Data?

OracleAS Personalization collects the data using Java calls provided by the REAPI. These calls add information 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 *stateful* Web applications (that is, an application that stores an identifier for each session)

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

The data temporarily stored in the REAPI cache is periodically flushed and sent to the appropriate RE. The session data is then used, combined with the customer’s historical profile data and predictive models, to generate recommendations. Finally, the RE instance periodically flushes the data to the Mining Table Repository (MTR) for sessions that have concluded or timed out. The OracleAS Personalization administrator can set configuration parameters to indicate what data is saved to the MTR. The data in the MTR is then used to build predictive models for future deployment via packages.

1.7 What Does OracleAS Personalization Do with the Data?

OracleAS Personalization uses custom profile data, data preparation specifications such as binning and item classification taxonomies to build predictive models. The data mining algorithms used by OracleAS Personalization analyze customer historical data resulting in the predictive model.

As more and newer customer profile data is collected and made available to Oracle Application Server, the accuracy of the recommendations generated by the models can improve. Hence, predictive models should be rebuilt whenever significant amounts of new profile data is added to the MTR. OracleAS Personalization provides the option to include the most recent *n* days of data from which to build a predictive model. This sliding window may, for example, specify the most recent 90 days.

The OracleAS Personalization administrator creates and manages schedules for building the packages, and for deploying the packages to the recommendation engines (REs). For ease of management and reliability reasons, recommendation engines with the same package are grouped together in recommendation engine farms (RE Farms).

See Also: Section 1.4, "How It All Works"

1.7.1 Predictive Models Built by OracleAS Personalization

OracleAS Personalization uses two algorithms: Transactional Naive Bayes (TNB) and Predictive Association Rules (PAR). For ratings-related recommendations, the model produced by TNB is used. For cross-sell related recommendations, PAR is used. See Appendix A for a description of the algorithms.

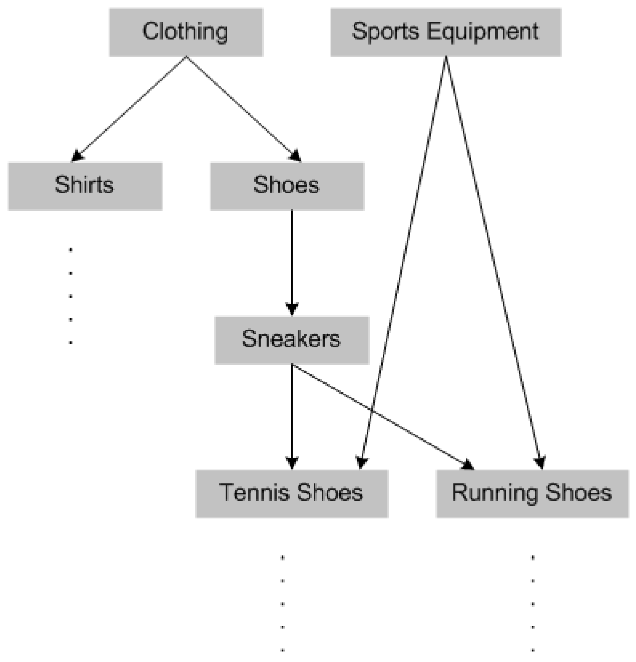
1.7.2 Defining and Using Taxonomies

A taxonomy, also known as concept hierarchy, defines classifications for a set of items, for example, products, Web pages, or documents. In OracleAS Personalization, an item can be a member of multiple categories, and each category can be a member of another category. For example, the catalog of a Web site using OracleAS Personalization has taxonomies with the most general category at the base (for example, "clothing"), and branching to increasingly specific categories (for example, from "clothing" to "shoes" to "sneakers" to "tennis shoes").

Items and categories can belong to more than one category and at different levels of the structure. For example, "tennis shoes" can be a category in "clothing" and also a category in "sports equipment."

The OracleAS Personalization taxonomy type is a direct acyclic graph (DAG), which can contain multiple top-level nodes, but there cannot be a path that connects a node's child back to the node itself. The different portions of the taxonomy can be disconnected too. The example in Figure 1–3 is a DAG.

Figure 1–3 Example of an OracleAS Personalization taxonomy



OracleAS Personalization also supports multiple taxonomies (different ways of organizing the items). Each unique taxonomy can be specified with a unique taxonomy ID. You can define each taxonomy according to how you want to retrieve recommendations (see Section 1.7.3, "Filtering Recommendations"). A taxonomy is implemented using a group of tables in the OracleAS Personalization MTR schema. The predictive models built in OracleAS Personalization use the taxonomies to generate category-to-category and category-to-item rules in addition to item-to-item rules.

1.7.3 Filtering Recommendations

In OracleAS Personalization, recommendations can be filtered to include or exclude items under a category level for a taxonomy. An application can filter recommendations using the `FilteringSettings` class in the REAPI. This class allows the filtering of categories and category subtrees through to the leaf items.

As an example of using the REAPI for filtering, consider the taxonomy illustrated in Figure 1–3. If you desire to obtain recommendations from the "Shoes" category, your Web application can use the `FilterSettings.setCategorySubTreeFiltering()` method to obtain recommendations among "Sneakers," "Tennis Shoes," "Running Shoes," and items in their subtrees.

See Also: `FilteringSettings` class in *Oracle Application Server Personalization Programmer's Guide* and `Oracle Application Server Personalization API Reference`.

2

The OracleAS Personalization Administrative UI

The Oracle Application Server Personalization (OracleAS Personalization) Administrative UI is a browser-based user interface that permits the OracleAS Personalization administrator to manage RE Farms and REs, schedule and deploy predictive model packages, produce reports, and otherwise manage OracleAS Personalization.

This chapter provides an overview of the Administrative UI and then illustrates how to use the UI to obtain recommendations.

2.1 Administrative UI Overview

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

```
http://<hostname>:<port>/OP/Admin/
```

where *<hostname>* is the name of the system on which Oracle Application Server is installed and *<port>* is the port number assigned to the OracleAS Personalization Administrative UI pages.

Note that the URL is case-sensitive.

The Web pages available in the Administrative UI are as follows:

- Login Page
- Home Page
- Farms Page
- Packages Page
- Schedules Page
- Reports Page
- Log Page

2.1.1 Login Page

Enter a user name and password for the MOR schema (a valid user name and password were established during OracleAS Personalization installation; if you don’t know what they are, ask the person who installed OracleAS Personalization).

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

2.1.2 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. 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 the various pages. The tabs are **Home**, **Farms**, **Packages**, **Schedules**, **Reports**, and **Log**. The tab pages organize OracleAS Personalization administrative tasks.

2.1.3 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 and give it a name. This also requires information established during installation and configuration of the database.

See Also: Configuration chapter in *Oracle Application Server Personalization Administrator's Guide*

2.1.4 Packages Page

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

An OracleAS Personalization 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).

Note: A package does not contain build schedule information. This information is in the MOR.

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 by name. If there is no MTR connection, you are prompted to create one.
- 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 — that is, between higher recommendations accuracy and faster recommendations.
  - New products: Specify whether you want OracleAS Personalization 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.

2.1.5 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 (see the next section).

Build Schedule

Use this page to create, edit, delete, and in general manage the building schedule 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. When a package is built, it is stored in the MOR, ready for deployment.

This page also gives you the option of going directly to scheduling deployment of the package (see the next section).

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.

2.1.6 Reports Page

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

- **Purchasing session report:** Reports, for a user-specified period of time, the total number of sessions, the number of sessions that had at least one purchase, and the percentage of purchasing sessions to total sessions.

  The total number of sessions includes both visitor and customer sessions. Sessions that had purchases include only customer sessions.
- **Recommendation effectiveness report:** Reports, for a user-specified time period, the number of recommendation requests served by the system, the number of recommended items clicked per recommendation request, and the number of recommended items purchased per recommendation request.

  This report includes *all* clicked and purchased items in its "Clicked" and "Purchased" columns, even those items that are not part of the recommendations.
- **Itemized recommendation effectiveness report:** Reports, for a specified time period, how many times the items have been recommended, how many times they have been clicked, and how many times they have been purchased.

  Entries in the "Product ID" column have the following format:

  <Product type> **:** <Product id>

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

2.1.7 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**.

2.2 Obtaining OracleAS Personalization Recommendations

You must deploy a package before you can execute a Java program that requests recommendations or collects data.

Therefore, before you can obtain recommendations you must:

1. Create a Recommendation Engine Farm.
2. Add at least one RE to the Farm.

3. Populate at least the minimum data required in the MTR tables. You can do this with the demo data that is loaded at the time of installation.
4. Schedule a build and deployment.

This chapter illustrates performing these steps.

Note that you cannot build a package unless there is data available; see Section 1.7.1, "Predictive Models Built by OracleAS Personalization" for more information.

2.2.1 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.
4. After filling in all the fields, click **Test Connection** to determine whether the database connection is successful, that is, to verify that the information provided can be used to make a successful connection to the desired schema.
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.

2.2.2 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.

2.2.3 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.

2.2.4 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.

2.2.5 Summary

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

Check the **Packages** page after a few minutes to see whether the package has yet built and deployed. The **Packages** page displays the status; refresh the page by clicking **Go**.

When the package has built and deployed successfully, you can use it to collect data and make recommendations using the Recommendation Engine API.

2.2.6 MTR Sample

Next, browse the contents of the Mining Table Repository (MTR) database used to build the predictive model. This is the pre-populated MTR that is installed when OracleAS Personalization is installed if you select that option. This pre-populated 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. Table 2–1 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.

Table 2–1 Sample MTR Database Table

CUSTOMER_ID	ITEM_ID	ITEM_TYPE	ATTRIBUTE_ID	BIN_VALUE	DATA_SOURCE_TYPE
2	264	MOVIE	1	1	2

Table 2–1 (Cont.) Sample MTR Database Table

CUSTOMER_ID	ITEM_ID	ITEM_TYPE	ATTRIBUTE_ID	BIN_VALUE	DATA_SOURCE_ TYPE
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: The data is all from customer with CUSTOMER_ID = 2.
- ITEM_ID: The ID numbers identify particular movies. Demographic information for a customer always has an ITEM_ID of 0.
- ITEM_TYPE: Refers to kind of item being rated; here the type is movie. Demographic information for a customer always has an ITEM_TYPE of NONE.
- ATTRIBUTE_ID: Values 1, 2, and 3 are attributes from purchasing, rating, and navigational data; values 4, 5, and 6 are from demographic data (the last three rows), where 4 = age, 5 = gender, and 6 = marital status.
- BIN_VALUE: Ratings were binned in 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 OracleAS Personalization schemas, see the *Oracle Application Server Personalization Administrator's Guide*.

3 Using the REAPI Demo

This chapter presents several exercises using the REAPI Demo.

Running the REAPI Demo accomplishes the following:

- You can verify that OracleAS Personalization was installed correctly by using the REAPI to get recommendations. To run REAPI Demo, OracleAS Personalization must be installed properly and the REAPI Demo must be configured to access RE. The demo data must also be loaded into the MTR.
- 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 OracleAS Personalization).
- You can experiment with the configuration so that you can learn how best to configure RE to suit your Web site.
- You can familiarize yourself with OracleAS Personalization's Java operations and see how the calls work and how they interact.
- You can view the source code to see how the program uses the REAPI calls.

Note: When you are finished using the REAPI demo, you should remove the accounts associated with it. Leaving the accounts can result in security problems.

For detailed information about the REAPI, see the *Oracle Application Server Personalization Programmer's Guide*.

For details about installing REAPI, see the *Oracle Application Server Administrator's Guide.*

In a production environment, 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.

The REAPI and the REAPI Demo are installed on the system where Oracle Application Server is installed.

3.1 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 OracleAS Personalization, choose the option to install the demo data into the MTR.

Next, use the OracleAS Personalization Administrative UI (described in Chapter 2) to perform the following steps:

1. Create a movies MTR connection, using the sample movies MTR database supplied with OracleAS Personalization.
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 a 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.

3.2 Start REAPI Demo

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

```
http://<hostname>/OP/redemo/
```

where `<hostname>` is the name of the system on which Oracle Application Server is installed. This will bring up the REAPI Demo user interface. The first page that appears is the **Welcome to OP REAPI Demo** page.

3.2.1 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, as shown in Section 3.3.2.

3.2.2 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 of 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.

3.3 Exercises

This section provides exercises that you can work through to get a feel for how the REAPI Demo works. The exercises cover the following tasks:

- Creating a proxy
- Collecting data
- Making recommendations
- A stateful exercise
- A stateless exercise
- Changing a visitor to a customer

3.3.1 Values and Results

To get meaningful results from these examples, you must enter 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 OracleAS Personalization.

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

3.3.2 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 OracleAS Personalization installation for host, port, and SID.
- **User name:** The user name for the database. Enter the RE user name that was specified during OracleAS Personalization installation.
- **Password:** The password for the RE user. Enter the password that was specified during OracleAS Personalization 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.)

3.3.3 Exercise 2: A Stateful 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 stateful Web application.)

A stateful 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**.

OracleAS Personalization then returns 10 or fewer recommendations for the user of this session. It will look something like Table 3–1, 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):

Table 3–1 Recommended Top N Items for User ID = 1

Type	Item ID	Rating
MOVIE	345	1.000
MOVIE	383	2.000
MOVIE	147	3.000
MOVIE	223	4.000

1. 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.

2. 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**.

   OracleAS Personalization then returns 10 or fewer recommendations for the user of this session. In this case, it returns something like what is shown in Table 3–2. Again, it displays the item type (MOVIE), the item ID (a number identifying each movie), and the rating.

Table 3–2 Recommended Hotpick Items for User ID = 1

Type	Item ID	Rating
MOVIE	360	1.000
MOVIE	370	2.000
MOVIE	116	3.000
MOVIE	83	4.000
MOVIE	18	5.000
MOVIE	20	6.000

1. 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:

Table 3–3 Rate Items for User ID = 1

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

1. 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.

3.3.4 Exercise 3: A Stateless 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**.

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

Table 3–4 Cross-Sell for Items for Hot Picks for User ID = 1

Type	Item ID	Rating
MOVIE	72	1.2785

3.3.5 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.

3.4 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 algorithms used by Oracle Application Server Personalization (OracleAS Personalization) to create models. Models are used to generate personalized recommendations. The two algorithms are

- Predictive Association Rules
- Transactional Naive Bayes

OracleAS Personalization automatically picks the best algorithm to for a particular type of recommendation.

A.1 Predictive Association Rules

The most familiar use of association rules is what we know as "market basket analysis," that is, 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, for example, 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, that is, rules that are based on many examples in the database. Support is the percentage of records that obey the rule, that is, baskets that contain both A and B.

Confidence: Next, determine which rules have high confidence, that is, 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.

A.2 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 (that is, 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 following table 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):

Customer	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":

Customer	"Hits"
Joe	bananas
Jim	apples
Jim	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. OracleAS Personalization transforms data into transactional format, if necessary, before building a package.

Glossary

This glossary explains terms used in the text and terms encountered in discussions related to personalization and data mining.

Admin UI (Administrative UI)

A graphical user interface that enables you to manage OracleAS 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.

Algorithm

See Recommendation Algorithms.

Category

A group of similar items. A category is an element in a taxonomy; an abstraction for a group of items or categories. In OracleAS Personalization, any item or category can belong to one or more categories but not to itself, directly or indirectly. 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

Cross-selling is done when a customer shows evident interest in a certain item and recommendations for other items are made to him or her. The cross-sell 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.

Data Source Types

OracleAS Personalization uses data mining data of the following types: navigational, ratings, purchases, and demographics.

Demographics

The particular demographic attributes of interest to OracleAS Personalization are listed below. They are stored in the MTR in the CUSTOMER table/view, which consists of the following fields.

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 a predictive model to one or more recommendation engines after the model has been built. A deployment also establishes the necessary connections between the recommendation engine and the MTR. The recommendation engine can run in a separate database instance from MTR and MOR if the site administrator chooses.

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*.

I-I (Item-to-Item)

"I-I" is encountered in some detailed error messages. It stands for Item-to-Item.

Interest Dimension

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

Mining Object Repository (MOR)

The MOR is the Oracle database schema that maintains mining metadata defined by the OracleAS Personalization data mining schema and provides 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 is a schema containing the data used for data mining. It contains all the data necessary to define and build a package. For OracleAS Personalization, 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.

OP

Oracle Application Server Personalization, also OracleAS Personalization.

Personalization Index

The relative degree of individualization desired in OracleAS Personalization'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 (Person-to-Item)

"P-I" is encountered in some detailed error messages. It stands for Person-to-Item.

Package Settings

The parameters that instruct the MOR algorithms how to build the predictive models. See Predictive Model and Predictive Model Package.

Predictive Association Rules Alogrithm

This is a recommendation algorithm used by OracleAS Personalization (see Recommendation Algorithms). This algorithm's characteristics are:

- Models are based on association rules
- Builds models and scores datasets inside the database
- The mined rules, which form the basis of the predictive models, are stored in database tables
- The rule consequents are combined and ordered using a ranking function
- Consequents with higher ranks are returned as recommendations

Predictive Model

A model is a compact representation of the knowledge or patterns found in a particular dataset. In OracleAS Personalization, it consists of a set of tables containing all the data necessary to make recommendations. See also Recommendation Algorithms.

Predictive Model Package

An object created using the Administrative UI that controls predictive model building. A predictive model package consists of the following:

- predictive models for Predictive Association Rules and Transactional Naive Bayes algorithms
- MTR synchronization settings
- customer data synchronization interval
- time-out sessions period

Predictive model packages are deployed to REs, perhaps via deployment to RE farms.

Profile

All the data collected about a customer from that customer's sessions. These data are: navigational, ratings, purchases, and demographic data. Profiles are stored in the MTR.

Rating scale

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

Recommendation Algorithms

OracleAS Personalization bases its recommendations on one of two algorithms: Predictive Association Rules and Transactional Naive Bayes. For fuller descriptions of these algorithms, see Predictive Association Rules and Transactional Naive Bayes, in Appendix A.

Recommendation Engine (RE)

The front end of OracleAS Personalization. It is a set of PL/SQL routines that run in a database instance. Through the 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 Java classes that enable a Web application written in Java to collect and preprocess data used to build OracleAS Personalization models and to produce recommendations from OracleAS Personalization.

Recommendation Engine Farm (RE Farm)

A group of REs where the REs have identical deployment of predictive model packages. This grouping improve scalability and reliability of applications using OracleAS Personalization.

Recommendations

The following are characteristics of recommendations made by OracleAS Personalization:

- Uses demographics, taxonomies, data from past sessions, current session data
- Performs efficient and scalable scoring
- Builds predictive models offline
- Handles both registered customers and anonymous visitors
- Are generated from predictive models, which are in turn generated from using the Transactional Naive Bayes or Predictive Association Rules algorithm

Schedule Item

An object created using the Admin UI that controls when predictive 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 confidence in a prediction. Score is also a prediction + confidence of that prediction. Confidence is expressed as a number.

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.

SID

See System Identifier (SID).

System Identifier (SID)

An identifier for an Oracle database instance. In OracleAS Personalization, 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 OracleAS Personalization context, this term refers to the structural organization of items in a company's catalog or site. Typically, the catalog or site or both has a hierarchical structure, with the most general category at the base (for example, "clothing"), and branching to increasingly specific categories (for example, from "clothing" to "shoes" to "sneakers" to "tennis shoes").

Items can belong to more than one category and to different levels of the structure. For example, "tennis shoes" can be a category in "clothing" and also a category in "sports equipment."

The structure of the OracleAS Personalization taxonomy is a DAG (direct acyclic graph), which can contain multiple top-level nodes. The different portions of the taxonomy can be disconnected. 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.

OracleAS Personalization supports multiple taxonomies. A taxonomy is implemented using a group of tables in the MTR (they are specified at installation time).

Transactional Naive Bayes

This is a recommendation algorithm used by OracleAS Personalization (see Recommendation Algorithms). This algorithm's characteristics are:

- 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

User (of OracleAS Personalization)

The user of OracleAS 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 OracleAS Personalization.

Index