ATG Repositories Data Model

Presenter's Name
Presenter's Title

Safe Harbor Statement

The following is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Oracle Training Materials – Usage Agreement

Use of this Site ("Site") or Materials constitutes agreement with the following terms and conditions:

1. Oracle Corporation ("Oracle") is pleased to allow its business partner ("Partner") to download and copy the information, documents, and the online training courses (collectively, "Materials") found on this Site. The use of the Materials is restricted to the non-commercial, internal training of the Partner's employees only. The Materials may not be used for training, promotion, or sales to customers or other partners or third parties.

2. All the Materials are trademarks of Oracle and are proprietary information of Oracle. Partner or other third party at no time has any right to resell, redistribute or create derivative works from the Materials.

3. Oracle disclaims any warranties or representations as to the accuracy or completeness of any Materials. Materials are provided "as is" without warranty of any kind, either express or implied, including without limitation warranties of merchantability, fitness for a particular purpose, and non-infringement.

4. Under no circumstances shall Oracle or the Oracle Authorized Delivery Partner be liable for any loss, damage, liability or expense incurred or suffered which is claimed to have resulted from use of this Site of Materials. As a condition of use of the Materials, Partner agrees to indemnify Oracle from and against any and all actions, claims, losses, damages, liabilities and expenses (including reasonable attorneys' fees) arising out of Partner's use of the Materials.

5. Reference materials including but not limited to those identified in the Boot Camp manifest can not be redistributed in any format without Oracle written consent.

Agenda

- Repository Data Model Definition
- Property and ID

Learning Objectives

At the end of this lesson you should be able to:

- Understand the Repository Model Definition file
- Create a repository definition file to define a repository
- Use XML File Combine to extend a repository definition
- View the combined repository definition file
- Define a repository id including a compound id
- Use a property tag to define repository properties
- Define require, unique and multivalued properties in a repository

Section 1:
Repository Data Model Definition

SQL Repository Setup

- Step 1: Create the repository definition file to be used by the SQL Repository.
  - A template XML file defines the repository item's descriptors and their attributes.
  - It describes the relationship between your SQL database and the SQL repository.
- Step 2: Configure a SQL Repository component.
  - SQL Repository is a Nucleus component.
  - The *definitionFiles* property points to the SQL Repository definition file.
- Step 3: Create the SQL database schema on your database server.
  - ATG provides tools to generate the preliminary data schema.

Repository Component

- A Repository component is required for each repository in your application.
- A Repository component is a Nucleus component.
- A SQL Repository is derived from the class:
  - Atg.adapter.gsa.GSARepository
- The next page defines the important properties for a repository component.

Repository Component Properties

Property	Explanation
definitionFiles	▪ Path of your XML definition file
dataSource	▪ datasource to be used to connect database ▪ OOTB: /atg/dynamo/service/jdbc/JTDataSource
repositoryName	▪ Name of the repository
XMLToolsFactory	▪ Used to parse repository definition files ▪ OOTB: /atg/dynamo/service/xml/XMLToolsFactory
transactionManager	▪ Used to manage transaction boundaries ▪ OOTB: /atg/dynamo/transaction/TransactionManager
idGenerator	▪ Used to generate unique ids ▪ OOTB: /atg/dynamo/service/IdGenerator
loadItemBatchSize	▪ Maximum number of items to load from the database at one time ▪ Default value is 200

Repository Definition File

- Each repository can be defined in one or more XML definition files.
- A repository definition file describes the repository Items in a repository.
- It maps table columns to item properties.
- It is referenced by the GSARepository component which represents the SQL Repository.
- If one or more XML files are defined with the same path in different configuration path directories, they are combined using XML combination rules.

Example: Repository Definition File

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>

<!DOCTYPE gsa-template PUBLIC "-//Art Technology Group,
Inc.//DTD General SQL Adapter//EN"
        "http://www.atg.com/dtds/gsa/gsa_1.0.dtd">

<gsa-template>
  <item-descriptor name="user">
    <table name="MY_USER" type="primary"  id-column-name="id">
      <property name="id" column-name="id" data-type="string"/>
      <property name="name" column-name="user_name"
          data-type="string"/>
      <property name="age" column-name="user_age"
          data-type="int"/>
      <property name="birthday" column-name="birthday"
          data-type="date"/>
    </table>
  </item-descriptor>
</gsa-template>
```

Major Elements of the Definition File

- The Definition file must have an XML declaration.
- All tags must be included in a gsa-template tag.

- The <item-descriptor> tag:
  - Define an item type.
- The <table> tag:
  - Add a table to an item type.
- The <property> tag:
  - Define a property of a repository item.

The <item-descriptor> Tag

- Each repository item type is described by one item descriptor, defined with the <item-descriptor> tag.
- Its major attributes are:

name	The name of this item descriptor, unique within the repository(required)
Cache-mode	The caching mode for this item descriptor
description	Optionally describes this item descriptor Default: value of name

The <table> Tag

- The <table> tag specifies an SQL database table that store properties of repository items defined by this item descriptor
- Its major attributes are:

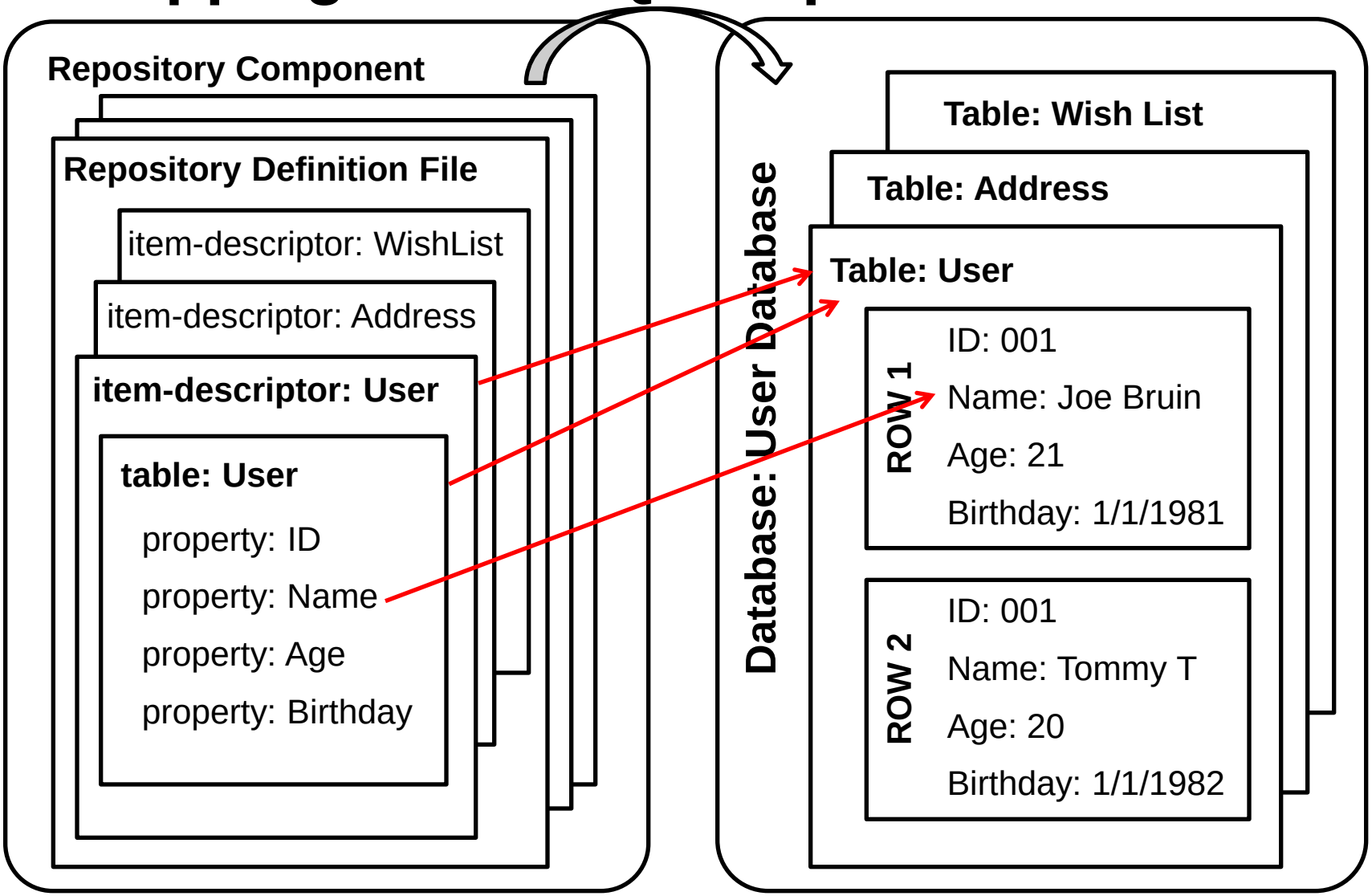
name	The table's database name
type	The table's type, one of the following Primary, auxiliary (default) or Multi
id-column-name(s)	The name or names of the database columns that correspond to the repository ID

The <property> Tag

- Each property tag corresponds to a property of the repository item.
- If the property tag is a child of the table tag, it represents a persistent property.
- If it appears outside the tag, it is a transient property and is not stored in the database.
- Its major attributes are:

`name`	`The property name (required)`
`data-type`	`The data type for this property, required unless item-type or property-type is set.`
`column-names`	`The column name or names in the SQL database` `Default: value of name`

Mapping of the SQL Repositories



XML File Combination

- The ATG platform combines same name XML definition files that appear along the configuration path into a single composite file in the runtime.
- It is controlled by an XML attribute xml-combine.
- The xml-combine attribute can be set to one of the following values to control Tag Combination:
  - replace, remove, append, append-without-matching, prepend, prepend-without-matching
- XML Files are recursive combination.

XML File Combination Example

```
<people>
 <person name="joe">
  <interests>
   <interest value="bass"/>
  </interests>
 </person>
</people>
```

```
<people>
 <person name="joe">
  <interests
        xmlcombine="append">
   <interest value="parenting"/>
  </interests>
 </person>
</people>
```

```
<people>
 <person name="joe">
  <interests>
   <interest value="bass"/>
   <interest
        value="parenting"/>
  </interests>
 </person>
</people>
```

Viewing the Combined File

- Open the component browser in dyn-admin, click the component's property definitionFiles.

CONFIGPATH
filename /atg/userprofiling/userProfile.xml

Source files
- /work/inexteam/ATG/ATG9.0/DPS/config/profile.jar/atg/userprofiling/userProfile.xml
- /work/inexteam/ATG/ATG9.0/DSS/config/config.jar/atg/userprofiling/userProfile.xml
- /work/inexteam/ATG/ATG9.0/DCS/config/config.jar/atg/userprofiling/userProfile.xml
- /work/inexteam/ATG/ATG9.0/B2CCommerce/config/config.jar/atg/userprofiling/userProfile.xml
- /work/inexteam/ATG/ATG9.0/DCS/CustomCatalogs/config/config.jar/atg/userprofiling/userProfile.xml
- /work/inexteam/ATG/ATG9.0/commerce/b2cblueprint/estore/config/config.jar/atg/userprofiling/userProfile.xml

System Id
(DTD name) http://www.atg.com/dtds/gsa/gsa_1.0.dtd

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE gsa-template SYSTEM "dynamosystemresource:/atg/dtds/gsa/gsa_1.0.dtd">

<gsa-template>

 <header>
  <name>B2CBlueprint International Related Profile Additions</name>
  <author>ATG</author>
    ...
```

Default Item Descriptor

- Each repository can have a single default item descriptor.
  - A repository's default item descriptor can be identified by setting the default attribute to true.
  - If a repository has only one item descriptor definition, it is the default.
  - If no default item descriptor is explicitly identified, the first item descriptor in the XML file is the default.
- When using repository API such as getItem, createItem without specifying an item descriptor, the default item descriptor will be used.
- Identifying the default item descriptor is recommended.

Example: Default Item Descriptor

- The user item-type is default for the following xml file.

```
<gsa-template>
  <item-descriptor name="user" cache-mode="locked"
                   item-cache-size="500" default="true">
    <table name="dps_user">
      <property name="userType" data-type="enumerated">
        <option value="investor" code="1">
        <option value="broker" code="2">
        <option value="guest" code="3">
      </property>
    </table>
  </item-descriptor>
</gsa-template>
```

Resource Bundle

- Java Resource Bundle supports a locale-specific resource:
  - Be easily localized, or translated into different languages,
  - Handle multiple locales at once,
  - Be easily modified later to support even more locales.
- SQL repository localization can localize:
  - Values of the display-name and description of both item descriptors and properties,
  - Category of properties,
  - Strings used for representing values of enumerated property types.

Define Resource Bundle

- Define a Resource Bundle using resourceBundle attribute.
- Can use the attribute tag to set the resource bundle at the property, table, or item descriptor.
  - A property uses its own resourceBundle attribute if it is set.
  - If not, it looks for a resourceBundle attribute set in its table tag, then for a resourceBundle attribute set in its item-descriptor tag.

```
<item-descriptor name="user" ....>
   <attribute name="resourceBundle"
        value="atg.userprofiling.ProfileResources"/>
...
```

Localizing Properties Display Name

- To localize labels in a repository editor(for example BCC), use localizable attributes:
  - display-name-resource: for display-name property
  - description-resource: for description property
  - category-resource: for category property
- For Example:

```
<item-descriptor name="user" display-name-resource="iUser">
   <attribute name="resourceBundle"
        value="atg.userprofiling.ProfileResources"/>
...
```

  - In resource bundle file ProfileResources.properties, it has entry:

```
iUser=User
```

Localizing Enumerated Types

- To localize enumerated types, use "resource" attribute on option tag:

```
<property name="emailStatus" ...
                data-type="enumerated" ...>
   <option resource="emailStatusUnknown" code="0"/>
   ...
```

- If you have useCodeForValue set to true, calling getPropertyValue does not return the localized property value.
- To display the localized value on a page, include the localized string in your page, using a Switch servlet bean to choose the proper value.

Section 1

Check Your Understanding

What are the steps for creating a repository?

Answer: Create the definition file, create the component and the database schema.

Section 1 
Check Your Understanding

What is a repository component?

Answer: A nucleus component that represents and controls the behavior of the repository.

Section 1 ✔
Check Your Understanding

What are the main tags in a repository definition file?

Answer: item-descriptor, table, and property tags.

Section 1 
Check Your Understanding

What is an item-descriptor tag?

Answer: It represents the item type. A sort of template of mappings between object properties and db tables.

Section 1 ✔
Check Your Understanding

What tag is used to map a database column to a repository item property?

Answer: A property tag.

Section 1 ✔
Check Your Understanding

Name a few ways in which xml files can be combined.

Answer: append, remove, replace, etc.

Summary

- To create a repository, create the definition file, the component and the database schema.
- A repository component is a nucleus component which controls the properties and behavior of the repository.
- A repository definition file defines the mappings between the repository items and the table/columns in the database.
- A repository definition file typically has the item-descriptor, table, and property tags.
- An item descriptor tag defines the repository item. A table tag specifies the database table. The property tag maps the items properties to the table column.
- Repository definition XML files can be combined in various ways.
- Resource bundles can be used to localize the definitions.

Section 2:
Repository Data Model Definition:

Property and ID

Repository ID Property

- The repository item table tag requires an id property to fetch the data from the table.
- ID property is a repository item identifier.
- ID property maps the table's primary key.
- The primary table must have id property.
- An example of an id property:

```
<item-descriptor name="user">
  <table name="user" type="primary" id-column-names="emp_id">
    <property name="id" column-name="emp_id"/>
  </table>
</item-descriptor>
```

Compound Repository IDs (1)

- ID property can span multiple database columns.
- One property can define all id columns:

```
<table name="doc" type="primary"
         id-column-names="folder_id,doc_id" >
   <property name="ID" column-names="folder_id,doc_id"
       data-types="string,int"/>
</table>
```

- Or define a property for each id column:

```
<table name="doc" type="primary"
       id-column-names=  "folder_id,doc_id" >
   <property name="folder" column-names="folder_id"
           data-type="string"/>
    <property name="document" column-names="doc_id"
           data-type="int"/>
</table>
```

Compound Repository IDs (2)

- The default id separator for compoung ids is ‘:’.
- GetItem method can use the compound repository id.

```
RepositoryItem user = rep.getItems("sales:102", "user");
```

- Use attribute id-separator of the tag item-descriptor to specify a different separator character.

```
<item-descriptor name="employee" id-separator="*">
  <table name="user" type="primary"
           id-column-names= "dept_id,emp_id">
     properties...
  </table>
</item-descriptor>
```

- The repository id in the example would be ‘sales*102’.

The <property> Tag

- Each property tag corresponds to a property of the repository item.
- Roughly each property tag corresponds to a column in the database table.
- If the property tag appears outside the table tag, it represents a transient property that is not persisted in the data store.
- Its major attributes are:

name	The property name (required)
data-type	The data type for this property, required unless item-type or property-type is set.
column-names	The column name or names in the SQL database Default: value of name

Types Of Properties

- Each property can be single valued or multi valued.
- Single valued properties can have a type of:
  - int,
  - string,
  - float,
  - double,
  - etc.
- Multi-valued properties can have a type of:
  - list,
  - set,
  - map.

Example: Types Of Property

- Single-valued:

```
<property name="lastName" data-type="String"/>
<property name="age" data-type="int"/>
```

- Multi-valued:

```
<item-descriptor name="user">
   ...
  <table name="userEmail" type="multi"
                  id-column-name="userid">
   <property name="emails" data-type="set"
     component-data-type="string" column-name="email">
   </property>
  </table>
</item-descriptor>
```

SQL Types and Repository Data Types

Reposiotry data-type	Java object type	Recommended SQL data type
string	Java.lang.String	VARCHAR, CLOB (Oracle)
int	Java.lang. Integer	INTEGER
float	Java.lang.Float	FLOAT (DB2, MS) NUMBER (Oracle)
double	Java.lang.Double	DOUBLE (DB2, MS) NUMBER (Oracle)
boolean	Java.lang.Boolean	NUMERIC(1) TINYINT (MS)
data	Java.util.Date	DATETIME (MS) DATE (DB2, Oracle)
set	Java.util.Set	NONE
map	Java.util.Map	NONE
list	Java.util.List	NONE

Enumerated Properties

- Enumerated item properties are string properties that are constrained to a predefined list of valid values.
- ATG supports two enumerated data types:
  - enumerated: Stores integer codes to the database.
  - enumerated String: Stores string codes to the database.
- Converting Integer Codes to Strings
  - By default, an enumerated property returns its value as an integer code.
  - You can configure an enumerated property so the repository converts the integer code into a string value by setting the useCodeForValue attribute to false.

Example: Enumerated Properties

```
<item-descriptor name="transaction">
  <table name="PRJ_TRANS" type="primary" id-column-name="ID">
    <property name="id" column-name="ID" data-type="string"/>
    <property name="amount" column-name="AMT" data-type="float" />
    <property name="transactionType" column-name="TRANS_TYPE"
                      data- type="enumerated">
       <option value="credit"   code="200"/>
       <option value="debit"    code="201"/>
       <option value="purchase" code="202"/>
    </property>
    . . .
  </table>
</item-descriptor>
```

The PRJ_TRANS Table

ID	AMT	TRANS_TYPE
101	10.34	200
102	11.35	201
103	24.31	200
103	15.10	201

Required and Unique Properties

- You can require that a repository item property is always set to a non-null value.

```
<property name="lastName" data-type="string"
            required="true" />
```

- Some repository item properties require a unique value.

```
<property name="login" data-type="string">
   <attribute name="unique" value="true"/>
</property>
```

Multi-valued Properties

- The SQL Repository implements one-to-many relations as multi-valued properties.
- It does not interpret the results according to any specific paradigm.
- This allows the application to apply whatever meaning you want to one-to-many relationships.
- The <table> tag for a multi-valued property must set the type attribute to "multi."
- The <property> tag for a multi-valued property must set the following attributes:
  - data-type: list/array/set/map
  - component-data-type/component-item-type: data type of each element

Operating on multi-valued properties

- The data type of the property must be set to:
  - Array,
  - Set,
  - Map,
  - List.

```
<property name="interests" data-type="set"
      component-data- type="string" column-name="interest"/>
```

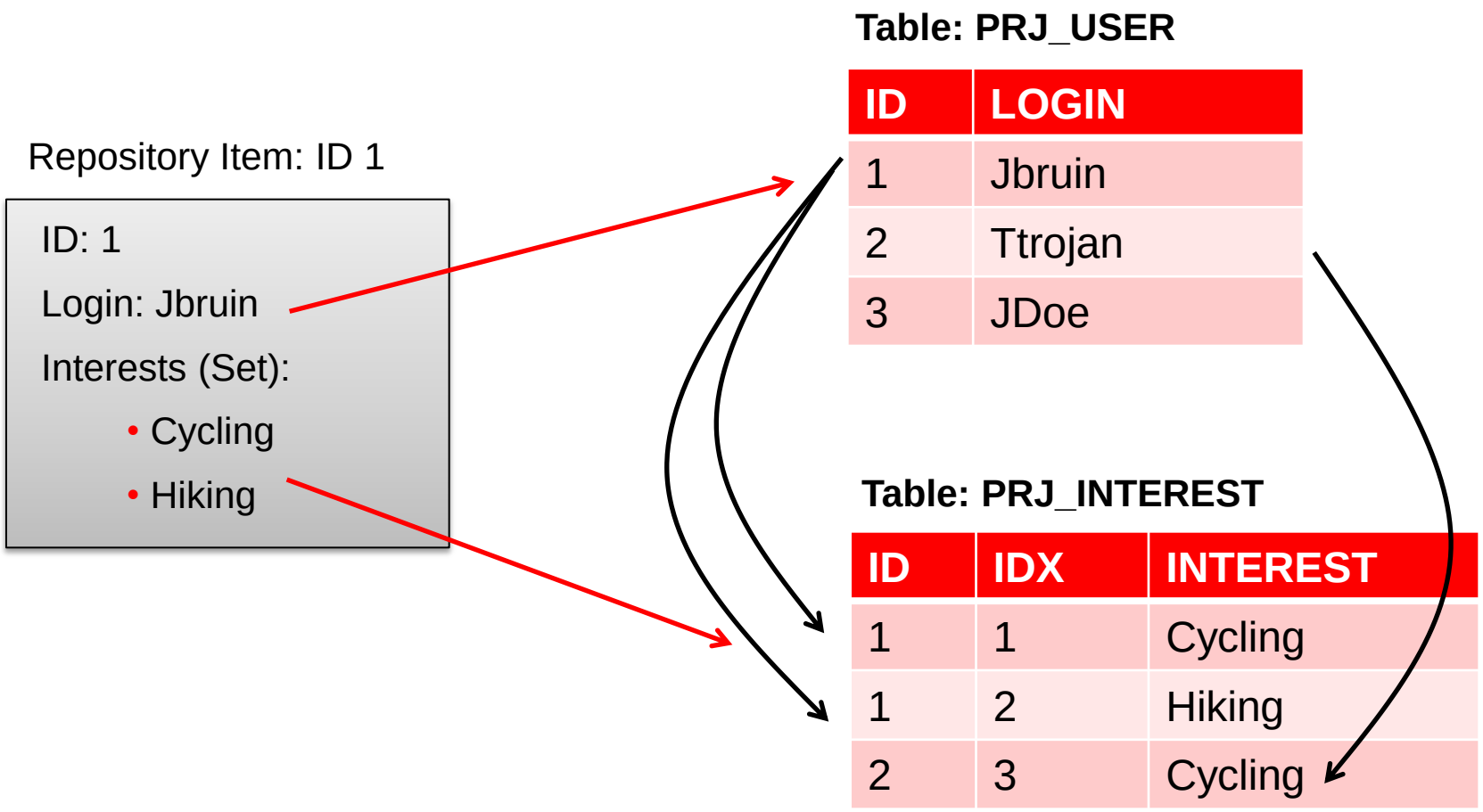
- Use the component-data-type to specify the data type of the value in the collection.
- Do not use the concrete implementation of the class. Instead use the interface of the class.
- Do not use in another set value. Copy the values into another collection and use that value instead.

Example: Multi-valued Properties

- A given user has many interests.
- We are representing the interests property in the item user as multi valued set property.
- The interests are located in another table called prj_interests.

```
<item-descriptor name="user">
  <table name="prj_user" id-column-names="id" type="primary">
    <property name="login" data-type="string"/>
  </table>
  <table name="prj_interest" type="multi"
        id-column-names="id"  multi-column-name="idx">
    <property name="interests" column-name="interest"
       data-type="set"  component-data-type="string"/>
   </table>
</item-descriptor>
```

Multi-valued property example



Section 2

Check Your Understanding

What is the property that uniquely identifies the repository item?

Answer: ID Property.

Section 2

Check Your Understanding

If your database table has a compound primary key, how is it represented in the item descriptor?

Answer: You can specify a compound key using a comma separated list.

Section 2
Check Your Understanding

Name a few scalar (single-valued) property data types.

Answer: String, int, boolean, float, etc.

Section 2

Check Your Understanding

If the property should only accept a specified set of values, how can you define them in the item-descriptor?

Answer: Using enumerated properties.

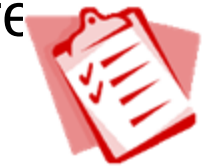
Section 2 
Check Your Understanding

Name a few multi valued data structures that can be represented as properties.

Answer: List, map, set, array.

Summary

- The Repository Id property is used as a unique identifier to refer to the repository item and to fetch the data from the data store.
- ID can be simple or compound, consisting of multiple database columns.
- A property tag can be single valued such as string, int, etc. It can also be multi valued such as list, set, map, etc.
- A property can be enumerated forcing the application to deal with a specified set of values that the property can accept.
- Required and unique attributes can be used to ensure data consistency.



Q&A

ORACLE IS THE INFORMATION COMPANY


ORACLE®