IKANALM

# Application Lifecycle Management for Oracle Warehouse Builder

IKAN ALM and VCR4OWB

## Table of contents

## Oracle Warehouse Builder (OWB)

Oracle Warehouse Builder (OWB) is a comprehensive tool for ETL (extract, transform and load), relational and dimensional modeling, data quality, data auditing, and full lifecycle management of data and metadata. With the Oracle Database as its metadata repository and transformation engine, OWB provides superior performance, security, and scalability.

OWB helps you with the design of your relational targets, mappings and process flows, with the deployment of relational database or dimensional objects and ETL mappings (generation of PL/SQL scripts). A mapping is a Warehouse Builder entity that describes the sequence of operations required to extract data from sources, transform the data, and load the data into one or more targets.

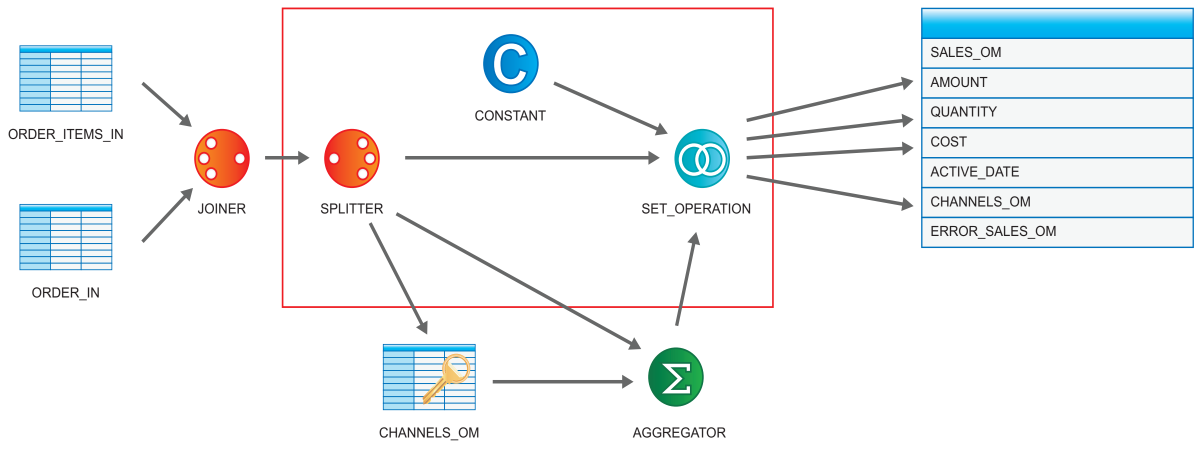
Mappings provide a visual representation of the flow of the data and the operations performed on the data. Warehouse Builder has 4 types of mappings: PL/SQL mappings, SQL*Loader mappings, SAP ABAP mappings and Code Template mappings.

In Oracle Warehouse Builder, you can execute two types of deployed objects: mappings and process flows. After you deploy process flows or mappings to your target system, they are available for execution from within the Control Center Manager. Additionally, process flows can be executed through Oracle Workflow.

The Warehouse Builder mappings can be executed individually each time you want to load or reload data or you can execute a process flow. The process flow will automate the order and dependency of the mappings execution.

In Warehouse Builder, you specify how to transform the data by designing mappings in the Mapping Editor.

A mapping is a Warehouse Builder entity that describes the sequence of operations required to extract data from sources, transform the data, and load the data into one or more targets. Mappings provide a visual representation of the flow of the data and the operations performed on the data. Warehouse Builder has 4 types of mappings: PL/SQL mappings, SQL*Loader mappings, SAP ABAP mappings and Code Template mappings.



Example of Mapping

## OWB Deployment Concepts

Deployment is the process of creating physical objects in a target location according to the logical objects in an OWB workspace.

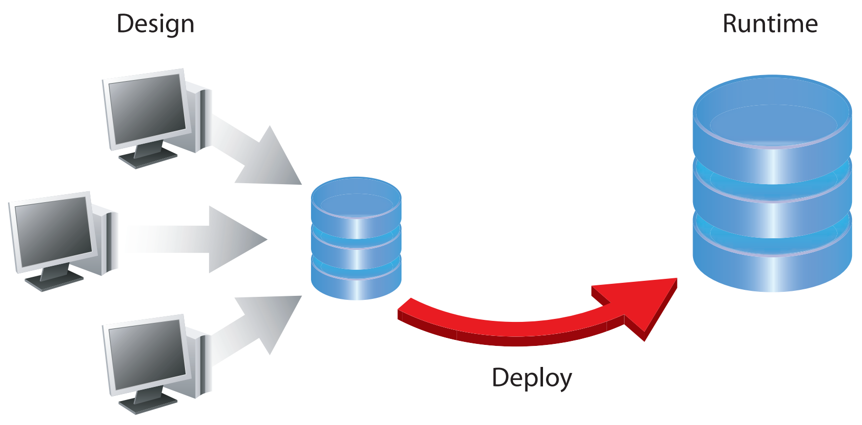
For example, when you create a table using the Design Center, the metadata for this table is stored in the workspace. If the table described in your design does not already exist in the database schema referenced by the specified location, then you must create the table by deploying it.

Similarly, after you design a PL/SQL mapping, you must generate code for it (which creates a PL/SQL package implementing the mapping logic), then deploy the generated code to the specified location, which loads the generated PL/SQL package to the referenced schema.

You can deploy objects from within the Design Center, or use the Control Center Manager. You can also use OMB*Plus commands to deploy objects.

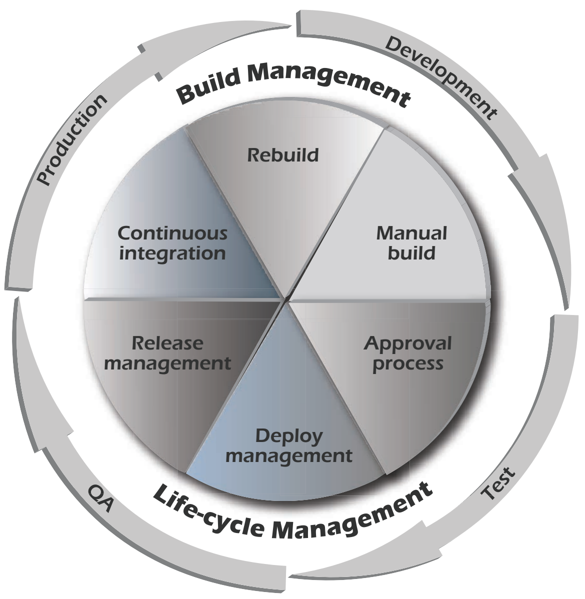
As soon as you define a new object in the Design Center, the object is listed in the Control Center Manager under its deployment location.

- Generate the PL/SQL, SQL*Loader, or ABAP script, if necessary
- Register the required locations and deploy any required connectors. This ensures that the details of the physical locations and their connectors are available at runtime.
- Transfer the PL/SQL, XPDL, SQL*Loader, or ABAP script from the Design Center to the Control Center.



Deployment process

# OWB and Application Lifecycle Management



Oracle Warehouse Builder provides a number of lifecycle features such as a Repository and related security, Impact analysis, data lineage and change propagation, the ability to handle multiple deployment configurations and scripting.

A standard ALM  practice in Oracle Warehouse Builder is to create a repository and one or more target schemas for each environment, where each environment has his own project definition, distinct target schemas per environment. Code is promoted by using MDL (Metadata Loader) export functionality and/or OMB*Plus scripting.

OMB*Plus, which is based on Tcl (Tool Command language) allows you to access all Repository features, allows you to create, change and deploy objects and mappings and supports import and Export between repositories.

However this approach demands a good OWB skills level and related experience. Such a process is complex and required good and constant communication between all involved parties, from developers over QA people and operations.

A normal OWB project has one Repository with several projects defined in it. The Repository uses single user locking, which makes it impossible to work with different developers concurrently on the same project.  A solution to overcome this is to use a Repository per developer (a "sandbox") and use export and import facilities to coordinate development and related changes. However OWB has no versioning API and is metadata driven.

A simple export of a single object exports not only the specific object, but also the dependent, related objects. An export of a single object like a mapping, by example, also exports dependent objects like tables, constants, schedules ... When you have a number of developers it is almost impossible to manage that process.  One option would be to work at project level and to just export, import a project or just operate as a black box.

VCR4OWB  however  provides the developer the possibility to work at a single object level. From within the OWB graphical interface, a developer can automatically do a "checkout", "commit", "export" or  "remove" and this at  every single object level. VCR4OWB also takes into account the dependencies and checks automatically, if an object has really changed, through a special DIFF function. Normally an object in the "sandbox" will always be different from the Repository as it will have a different time stamp, user name orRepository name. The VCR4OWB DIFF function takes abstraction of these non-changes and transfers only the objects that have really changed.

And with direct access to not only developing and testing platforms but also to the production environment, a great responsibility rests on each involved stakeholder, especially developers.

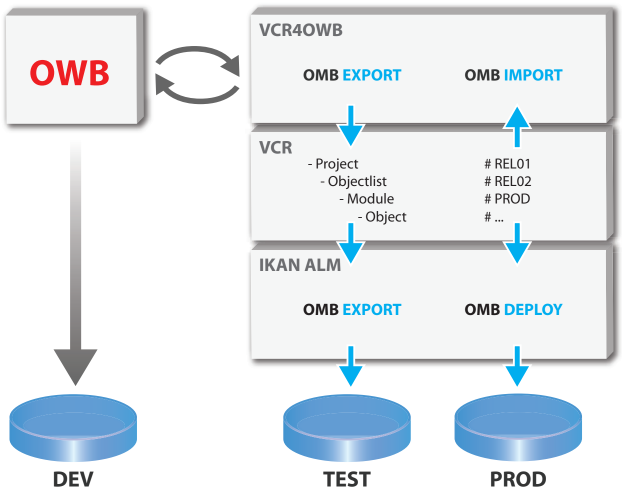
Not to mention ALM good practices, where separation of duties and a repeatable process is key. Last but not least, a standard environment doesn't just consist of OWB, but can also involve other tasks or processes. An example of such other tasks are all the database related tasks.

## IKAN ALM and VCR4OWB: Professional Application Lifecycle Management for OWB

IKAN ALM with VCR4OWB offers a complete, automated solution for OWB related Application Lifecycle Management (ALM).

## VCR4OWB, independent versioning for OWB

First of all VCR4OWB allows you to version the OWB objects into CVS, an open Source Versioning Repository. A developer can select individual objects (mapping, table, ...) or he can select a whole list of objects (mappings, modules, project) at the same time. VCR4OWB, offers concurrent versioning at object level (which is the lowest entity level of OWB) and uses "standard" xml files as format. Intelligent diff- parsing helps to avoid useless version changes, like just a timestamp change. VCR4OWB complies with the OWB standards and makes use of the OWB Experts technology.



**VCR4OWB consists of a JAVA class with five (5) functions:**

### 1. CHECKOUT

Does a checkout of the given VCR-module/object into a given path (sandbox) and writes a list of all elements checked out into a given text file.

### 2. COMMIT

Does a commit of selected elements named in a given text file from a given path (sandbox) into a given VCR-module/ object.

### 3. EXPORT

Similar to CHECKOUT but without the VCR-information. So elements from foreign projects can be mixed into the default project.

### 4. REMOVE

Does a delete of selected elements in a given text file in the given path (sandbox) and then commit to set them "removed" in the VCR.

### 5. STATUS

Does a VCR-status command in the given path (sandbox) and shows an html-list of all elements in the path to give a visual compare of the situation in OWB and helps solv-ing conflicts.

This JAVA class can be used in OWB with the "Experts technology" to completely integrate this VCR-function in the development environment of OWB. Sample "Experts" are delivered on how to use it or a customer can develop or write his own Expert.

## IKAN ALM: Lifecycle, Build, Deploy and Approval Management

Once all OWB and non-OWB objects are versioned and available, IKAN ALM can help you to Build and Deploy your application. Deployment can happen over a customizable Lifecycle: you can define as many levels as you want, whereby each level can consist out of one or more physical environments.

In addition you can also define pre- and post-approvals to each level. IKAN ALM has as benefit that it can integrate the OWB processes with other processes or tools in order to come to one global "release" embedding OWB and non-OWB elements. Example of non-OWB processes are the database related tasks.

### Build

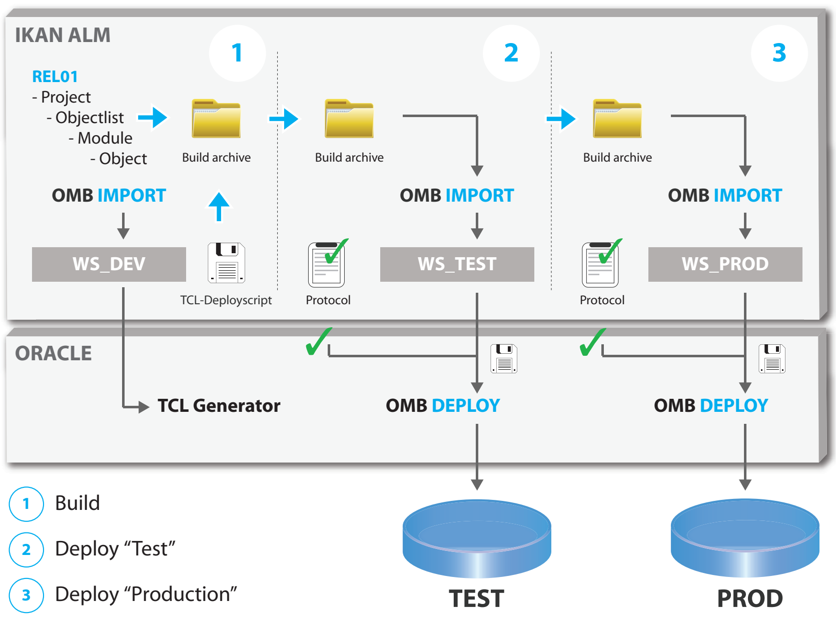
The objects are retrieved from the VCR and imported into the local OWB instance (build machine). Depending on the objects in OWB, a Tcl-script is created dynamically, which describes all steps and dependencies for the later OWB-deploy (writing PL/SQL into the database). This script is parameter driven, so it can be used later with different parameters for the rollout to several environments.

### Deploy to Test & Deploy to Production

Since all necessary files and scripts are packaged with the build, the deploy runs straight in two steps: As a first step an import of the metadata into the deploy OWB-repository is done and as a second step, the Tcl-script to deploy everything into the target database is executed. This is our possibility to use OWB capabilities for deploy.

The dynamic Tcl-creation is handled by a separate process. The only task left is to import the OWB metadata and triggering the creation and execution of Tcl-scripts.

In the current release, only full builds are supported. In the future we will provide the possibility to do also incremental builds and deploys.



## IKAN ALM and VCR4OWB benefits

- ✓ Platform independent web application
- ✓ Ergonomic user interface for intuitive operation
- ✓ Detailed, customized process design of all server, build, and Deploy tasks
- ✓ Extensive reporting database with open interfaces
- ✓ IKAN ALM support various version control tools (Eg, CVS, IBM ClearCase, MS Visual SourcesSafe, PVCS, SubVersion ...)
- ✓ Currently only CVS is supported for OWB.
- ✓ Generic interface for integration of URL-based tracking systems
- ✓ Full integration into existing work environments and tools
- ✓ Full command line interface
- ✓ JAVA interface VCR4OWB for the integration of OWB Experts
- ✓ Build-controlled processes, independent of the versioning tool
- ✓ Reuse of existing Build and Deploy scripts possible
- ✓ Complex, individually configurable role model for the control of any inspection and approval stages to regulate the access skills
- ✓ Reliable feedbacks on mail and web interfaces
- ✓ Distributed, incremental, multi-platform Build and Deploy
- ✓ Definition of project dependencies
- ✓ Support for multiple, parallel life cycles per project (eg: for release developments, production, maintenance and emergency operations, etc.)
- ✓ Support for parallel or sequential rollout (if necessary at the same time)
- ✓ Easy roll back to earlier release versions
- ✓ Integration with existing security systems and policies (E.g., Open Directory, LDAP, etc.)

More info on IKAN ALM: http://www.ikanalm.com
More info on VCR4OWB and dynamic Tcl  script: http://www.minerva-softcare.de

**IKAN Development (Belgium)**
Kardinaal Mercierplein 2
2800 Mechelen, Belgium
Tel. +32 15 797306

**IKAN Development (France)**
3, Rue du Général De Gaulle
28700 Aunay-Sous-Auneau, France
Tél: +33 2 37 25 31 22

info@ikan.be
www.ikan.be