



# Oracle 9i Warehouse Builder

Architectural White paper

January 2003

# Table of contents

- INTRODUCTION ..... 3**
- OVERVIEW ..... 4**
  - THE DESIGN COMPONENT ..... 4
  - THE RUNTIME COMPONENT ..... 5
- THE DESIGN ARCHITECTURE..... 6**
  - THE REPOSITORY ..... 6
  - THE METABASE ..... 6
  - THE API AND SDK LAYER..... 7
  - IMPORTING METADATA ..... 8
  - DEPLOYING METADATA AND CODE ..... 8
  - THE CLIENT TOOL..... 9
  - THE METADATA REPORTING ENVIRONMENT ..... 10
- THE RUNTIME ARCHITECTURE..... 11**
  - THE ETL ENGINE ..... 11
  - DATA STORAGE AND QUERYING FACILITIES ..... 12
  - PARALLEL PROCESSING BY DEFAULT ..... 13
  - DATA QUALITY IN THE ENGINE ..... 13
  - RUNTIME INFORMATION AND REPORTING..... 13
- CONCLUSION ..... 15**

---

## Introduction

Oracle 9i Warehouse Builder is the only enterprise Business Intelligence integration design tool that manages the full life-cycle of data and metadata for the Oracle9i database. Leveraging Oracle 9i's Business Intelligence features and scalability, Warehouse Builder allows designers and developers to design, instantiate, manage and maintain data warehouses for the enterprise or for the department.

The process to design and develop a business intelligence solution is complex. To select the right tool and vendor plays a great part in easing this process. This white paper describes the Oracle Warehouse Builder architecture and explains why using this tool is ideal for customers choosing an ETL and design environment for their business intelligence projects.

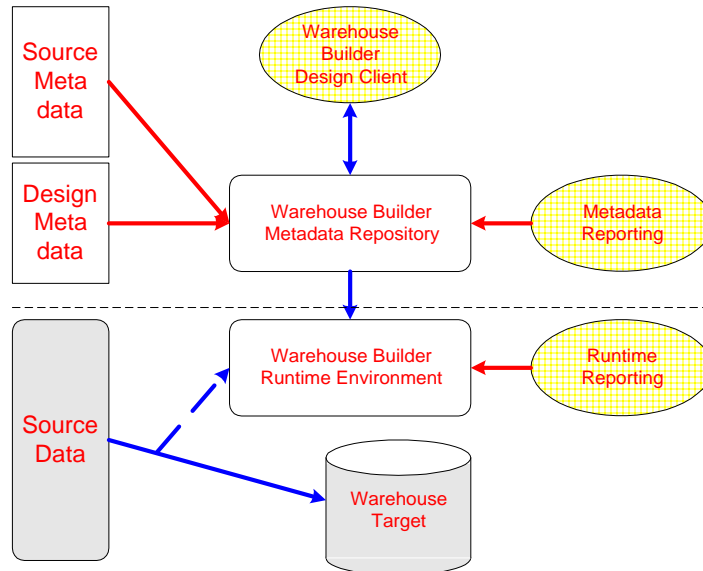
The core to all business intelligence solutions is consolidation and integration of data. However, organizations generally tend to buy and use a variety of tools from different vendors. A second integration problem emerges when they try to integrate their metadata between these disparate tools. Oracle's value proposition is to provide an integrated solution; not to integrate tools for each individual organization. Accordingly organizations only face the task of integrating their data.

Why is Oracle Warehouse Builder so important to the business intelligence solution? The answer is found in the importance of metadata for a number of different users in the organization. The ETL tool should, and Oracle Warehouse Builder does, play an important role in the overall metadata strategy of the organization. Instead of integrating tools by trying to synchronize metadata stores, Oracle Warehouse Builder owns most of the query tool information and allows query and OLAP tools to use this metadata. Instead of integrating tools and information Oracle Warehouse Builder provides an integrated solution out of the box.

## Overview

Warehouse Builder is a repository-based tool for ETL and data warehousing. The basic architecture comprises two components, the design environment and the runtime environment. Each of these components handles a different aspect of the system; the design environment handles metadata, the runtime environment handles physical data.

The metadata component (the top half in the figure) revolves around the metadata repository and the design tool. The data component (the bottom half in the figure) revolves around the runtime environment and the warehouse database.



### The design component

The Warehouse Builder design component consists of a highly scalable metadata repository that is stored in an Oracle database and a set of client design and reporting tools written in Java or HTML. Using these tools metadata can be viewed and manipulated.

Creating metadata is a design activity that uses the client tool to design objects, processes and jobs in the editors. This interactive way of creating metadata is typically used to design a new system. Warehouse Builder supports the design of relational database schemas, multi-dimensional schemas, ETL processes and End User tool environments through the client.

Source systems play an important role in any ETL solution and instead of creating this manually, Warehouse Builder provides integrated components that import the relevant information into its repository.

One of the strengths of the architecture is the so-called life-cycle management that allows metadata to be updated based on changes in the source systems. Warehouse Builder then facilitates propagating these changes to the ETL processes and the target systems.

To ensure the quality and completeness of the metadata in the repository Warehouse Builder provides extensive validation within the repository. Validation helps to keep a complex system that is created by multiple users in an accurate and coherent state.

To further aid in the development and evaluation of the metadata, a web based metadata reporting environment is available. The reporting environment allows developers and business users to browse and investigate system elements without using the design tools. A very important component of this reporting environment is the Impact Analysis capabilities allowing the identification of the impact of changes throughout the system before they are made. Impact Analysis reporting allows for better control on changes and better planning for the implementation of these changes. The opposite capability, tracing back where data originated from, is called Data Lineage reporting and is also provided in Warehouse Builder.

## The runtime component

Once the user has designed the ETL system on a logical level, he needs to move it to the physical database environment. Before this can be done, information about the database environment is added (configuration) to the logical design. After the configuration is complete, code can be generated.

Warehouse Builder generates extraction specific languages<sup>1</sup> for the ETL processes and SQL DDL statements for the database objects. The generated code is deployed, either to the file system or into the database.

Performing the ETL functions means running the deployed code in the database via for example Oracle Enterprise Manager. The ETL process then pulls the source data into the target database<sup>2</sup>. This can be a staging area, and operational data store, a warehouse or any other schema. Note that the code sections external to the Oracle database are executed in their respective environments. ABAP code to extract from SAP systems is for example run in the SAP environment.

To allow reporting on data loads, the code generated by Warehouse Builder contains audit routines. These routines write information about the ongoing load into the Warehouse Builder runtime tables. Information captured while running the code can include the number of rows selected, inserted and updated. If an error occurs while transforming or loading data, the audit routines report the errors into the runtime tables. To allow easy access and convenient reporting on this runtime information, Warehouse Builder provides the Runtime Audit Browser.

Dependency management and scheduling is provided by a close integration with specific Oracle tools. Oracle Enterprise Manager (OEM) is Oracle's scheduling and database management tool. Warehouse Builder creates jobs in the OEM repository, which can be scheduled and monitored along with other database activities. Through the interaction with Oracle Workflow (OWF) the Warehouse Builder user can create full-blown processes for dependencies between the ETL processes including notifications.

---

<sup>1</sup> Languages include: SQL\*Loader control files for flat files; ABAP for SAP/R3 extraction and PL/SQL for all other systems.

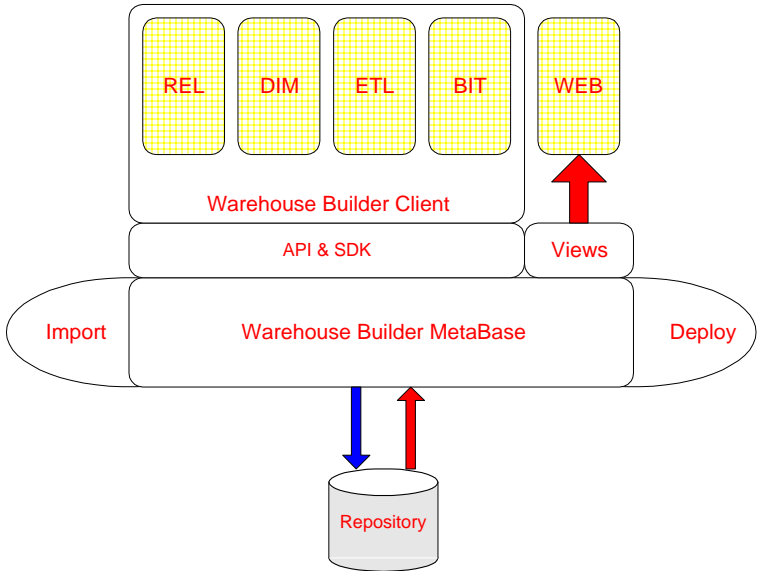
<sup>2</sup> Supported target databases for Warehouse Builder are Oracle 8i EE and Oracle 9i EE.

# The design architecture

Openness, productivity and reliability are keywords in defining a warehouse design platform. This chapter will discuss these characteristics of the Warehouse Builder design architecture. After the high level overview showing all building blocks for the Warehouse Builder architecture this chapter will detail the components of the design architecture.

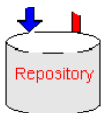
The central concept in the design architecture is a repository stored in an Oracle database. This repository, which is managed by the Warehouse Builder MetaBase, provides services to all producers and consumers of metadata. The graphical representation of the architecture shows that the MetaBase is the hub distributing and receiving metadata.

Through APIs the client tool provides graphical editors that manipulate the metadata in the repository via the MetaBase services. The import and deploy services are extensions to the MetaBase and act as interfaces to the world outside the MetaBase. To enable SQL access to the MetaBase public views are provided which are used by the metadata reporting environment. Each of these components as well as their main interactions will be discussed in detail in the following paragraphs.



## The repository

The repository is a relational structure holding data elements at a granular level. It stores data, which is aggregated in the MetaBase to represent objects to the users. The repository is in essence a data structure providing highly scalable data storage in an Oracle database. Since the repository is running on the Oracle enterprise edition database, openness, scalability and reliability are guaranteed.



## The MetaBase

The core functionality of the repository is separated from the database and is called the Warehouse Builder MetaBase. This component is the actual hub within the metadata management architecture delivering the bulk of the services required by the developer.



The MetaBase enables a scalable solution that allows for custom activities as required by the Warehouse Builder client design components. Two very important components related to the MetaBase are the import and deploy services that will be discussed separately. The other main services are discussed below.

### *Object storage and retrieval*

Objects are stored in the database. However, to store and retrieve them the MetaBase allows for a coordinated and controlled mechanism. This is decoupled from the database mechanism to allow for a stand-alone service of the MetaBase.

Warehouse Builder knows the concept of a First Class Object (FCO), which is a group of objects that is combined to one meaningful object for the user (e.g. a business component like a table is an FCO however; the table has multiple Second Class Objects (SCO) such as columns and constraints). The MetaBase ensures retrieval and storage of FCOs.

### *Multi-user and locking service*

In order to retrieve and then allow editing (e.g. write changes back to the repository) an object needs to be locked. Since this object (an FCO) consists of multiple components the MetaBase will need to lock all of the SCOs in order to ensure retrieval and editing of one object at the time. Since Warehouse Builder is multi-user, this locking is more complex. The first user to open an object will acquire the read/write lock on the whole FCO. All consecutive users will only get a read lock. The MetaBase will allow refreshing of this view after the user with the write privileges has committed his changes. The commit is translated by the MetaBase into a database commit.

### *Framework support for business logic*


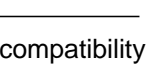
All services supplied by Warehouse Builder are rooted in the MetaBase, which provides a framework on top of which the business logic is layered. This business logic interprets the objects retrieved and stored by the MetaBase.

The main functions for the business logic are core to supporting Warehouse Builder generation and validation services. Validation allows verification of the metadata to allow for early reporting on mismatches or errors. Generation is a service that compiles the code and delivers the code.

## The API and SDK layer

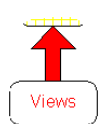
In order to allow the client tools to connect and talk to the MetaBase, a SDK (Software Developers Kit) and an API layer is provided. To allow for read-only SQL access, a set of public views is provided.

### *Public APIs and scripting*

This set of public APIs<sup>3</sup> allows users of Warehouse Builder to manipulate the repository in a  controlled manner without the user interface. Implementing the APIs is a phased project and in the initial release of OWB, the structural APIs are published. The  APIs are fully documented Java APIs. Since they are public backward compatibility is guaranteed.

Based on the APIs, Warehouse Builder provides users with a scripting language in which metadata manipulation commands can be executed from a command line interface. Scripting supports direct actions on the repository as well as the execution of batch scripts. This mechanism provides an ideal way of applying batch updates to a repository. The scripting language is based on the TCL scripting language.

### *Public views*



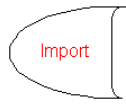
In order to provide read-only access to the repository via SQL, Warehouse Builder provides public views, both on the design and the runtime repository. The Warehouse Builder Browser uses these views to report metadata to end-users in a secure and efficient way. To build their own reporting environment on the repository users can

<sup>3</sup> To be released with version 9.0.3.x of the product.

utilize these views. The views are documented and backward compatible across releases.

## Importing metadata

Retrieving metadata from external systems is a major component of building a data warehouse.



In order to load data from a source system, an ETL tool must understand at least some of the metadata in that source system. Warehouse Builder has a unique architecture that allows detailed information to be extracted and reconciled for the source system.

From an architecture standpoint, Warehouse Builder applies two different types of metadata integration solutions.

Point-to-point solutions are built to extract metadata from specific systems adding intelligence to the extraction specific to these systems. An example of this would be the SAP Integrator solution<sup>4</sup>. This integrator is specifically designed for SAP R/3 systems and accesses the Business Object Repository of the SAP system via Remote Function Calls (RFCs). This allows Warehouse Builder to read the business logic of the SAP system and extract that information into the repository. Warehouse Builder uses this information to generate source specific extraction code (ABAP in the case of SAP).

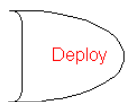
Another form of point-to-point metadata import (and export) is the import of Warehouse Builder's own metadata through Metadata Loader (MDL). MDL allows for metadata import from other Warehouse Builder repositories. This import capability is highly flexible and can even be used to patch existing metadata.

The second solution is the usage of standards based solutions. These can be internal or external standards. The most important standard and source for Warehouse Builder is the Oracle database catalog<sup>5</sup>. Other standards that Warehouse Builder supports are of course OMG-CWM (Object Meta Group – Common Warehouse Method) and ODBC.

A unique out of the box feature in Warehouse Builder is the metadata reconciliation from the sources with the repository information. This allows designers to add or alter the repository metadata in a controlled and predictable fashion. Warehouse Builder shows the impact on the metadata via a report and allows an assessment of the impact all the way to the target. This way the designer can estimate the impact and then choose to do propagate the changes in a planned timeframe.

## Deploying metadata and code

Deployment moves the design into a tangible warehouse environment. In that sense, deployment is the bridge between the design architecture and the runtime architecture.



Warehouse Builder allows developers to deploy a system to various environments. Depending on the scope of the generated code, deployment of an entire module can be done to multiple target systems. The same generated code can be deployed to a test and a production environment by simply changing the environment information. Individual objects as well as entire systems can be deployed, enhancing the granularity and flexibility of the deployment service.

<sup>4</sup> Other point-to-point solutions include a flat file integrator, Oracle Applications integrator, Oracle Designer and CA ERWin import. The latter two can be imported into target modules as well.

<sup>5</sup> Transparent gateways use this standard via the heterogeneous connection on the database, making the Oracle catalog the standard for relational systems access for Warehouse Builder.

### *Life-cycle management*

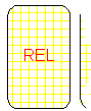
A unique feature that Warehouse Builder provides is the management of changes on the target database as a result of changes in the metadata definitions. Instead of dropping objects causing all sorts of issues in the warehouse environment, Warehouse Builder provides generation of update scripts. Before applying any of these updates, the designer can view the provided Impact Analysis report and choose to postpone or apply the changes. This reporting leads to a planned and easy upgrade of the warehouse without massive downtimes and reload operations, making Warehouse Builder's architecture unique within the ETL tool space.

### The client tool

Warehouse Builder contains a number of design tools to complete the design of the entire warehouse solution. Relational, multi-dimensional and Business Intelligence tool design are covered next to the core functionality in the ETL designer. The entire client software is written in Java. For more information on the client tools and capability, please look at the Warehouse Builder User's Guide or the overview white paper. The following paragraphs will summarize the capabilities of the client tool.

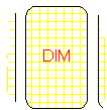
#### *Relational design*

The relational schema designer allows the warehouse designer to create relational schemas. The main objective of this designer is to allow the developer to design 3<sup>rd</sup> normal form schemas and allow for different ways to store data in the warehouse environment. A 3<sup>rd</sup> normal form ODS (Operational Data Store) can therefore be modeled in Warehouse Builder as easily as a dimensional structure. The designer also allows the developer to view source and warehouse schemas, including relationships, in a schematic view.



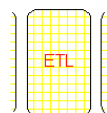
#### *Dimensional design*

The dimensional designer specifically enables easy design of multi-dimensional structures. It allows for easy creation of multiple hierarchies per dimension, multiple shared levels and easy linkage to the fact tables. Dimensions can have assigned roles and can be linked multiple times to the same fact. A fact editor provides a wizard driven method to create facts based on the dimensions available in the repository. Warehouse Builder provides automatic key and index creation on the fact foreign keys to allow star query transformation on facts generated from the tool.



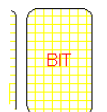
#### *ETL design*

The main function of Warehouse Builder is the design of ETL processes. In this design Warehouse Builder provides a separation between the physical implementation of the process and the business logic designed in the logical view of this process. ETL designers will first design the logical view of the process by specifying the business logic in a graphical picture. This logical view is configured and translated into a physical implementation. Specifying the configuration will dynamically influence the generated code. For example, when adding configuration and stating that the deployment environment is an Oracle 9i database instead of Oracle 8i, different code will be generated making use of specific ETL capabilities in the Oracle 9i database. Alternatively different configurations can be applied to different environments (development and production).



#### *Business intelligence design*

The business intelligence designer enables Warehouse Builder users to create the metadata for the Oracle Discoverer End User Layer (EUL) and populate it from Warehouse Builder. It allows specification of properties for the EUL objects and derives the metadata for the objects from the objects in the Warehouse Builder repository. For example a dimension object can become a folder object and inherits the hierarchy as an item class. All attributes in the dimension become the items in the folder.



These client capabilities make Warehouse Builder a flexible design and ETL tool generating optimized code for the Oracle environment and providing tight integration.

## The metadata reporting environment

In order to allow easy access to the important information in the Warehouse Builder repository a separate mechanism is added to view this metadata in comprehensive reports. Oracle Portal is the delivery platform providing reports with browsing capabilities in a user friendly and secure environment.



Warehouse Builder metadata reporting is browser based and inherits Portal security. Reporting is split into several portlets, including administration of the browser environment and the actual metadata browsing portlet.

These metadata reports all run on top of the public views, which means that all the information shown in the browser reports can be accessed through other tools. The reports are easily extendible, allow for custom information to be placed in them and can be combined with Discoverer portlets to create a comprehensive warehouse portal for administration and end users.

All of these reports can be launched from a browser without installing any Warehouse Builder software on the client computer. The reports can be launched from the client tool as well. Combining the information from the graphical Lineage and Impact Analysis reports, the designer of the warehouse has a powerful tool to quickly adapt the design of the warehouse to the ever-changing environment.

## The runtime architecture

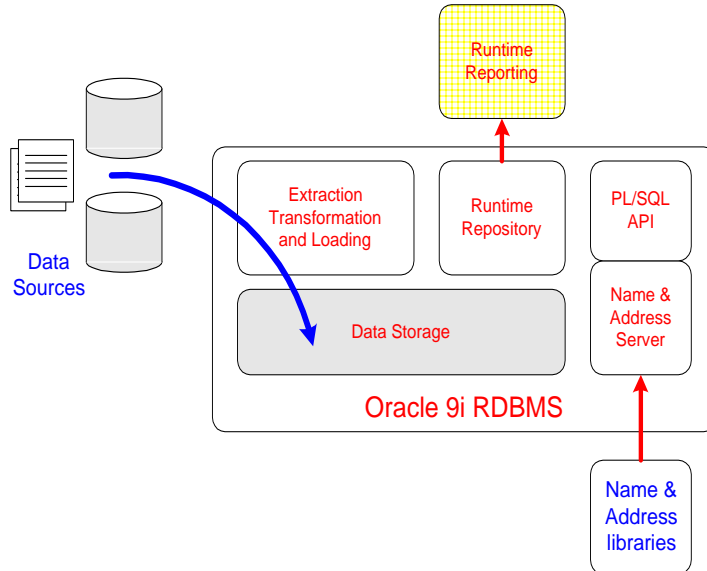
The runtime architecture of a warehouse system is as important as the design. Reliability, scalability and openness are keywords in defining an ideal warehousing runtime platform.

The graphical overview of the architecture shows that the Oracle 9i database is the platform on which all Warehouse Builder specific components run.

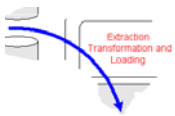
This is one of the major differences with many ETL tools in the market.

Warehouse Builder does not have its own external engine that runs ETL jobs. The jobs (generated code) run in the Oracle 9i database engine. This way the ETL process is brought to the data, not vice versa. The other advantage of this close alignment with the

database engine is the inherent scalability and reliability of the platform. Leveraging the ETL features built into the Oracle database, Warehouse Builder can provide organizations with code optimized for running on their Oracle database platform, reducing the need for additional tuning and training.



## The ETL engine



Warehouse Builder generates code that leverages many features of the Oracle database. The database performs dual roles in the runtime architecture. This paragraph will focus on the ETL engine capabilities and usage of this within Warehouse Builder code. The other role, data storage and retrieval engine, is covered in the next paragraph.

### Extracting data

A typical ETL process consists of three stages. The first is obviously the extraction of the data from the source systems. The architecture for Warehouse Builder is designed to allow extraction from heterogeneous platforms (into Oracle using various methods running as much code as possible on the Oracle target platform – re-word).

Warehouse Builder calls several utilities to extract data from source platforms. For files the default method is to use SQL\*Loader, the bulk loader provided with the Oracle database. SQL\*Loader allows the fast extraction from flat files in either direct path mode or in a more controlled, conventional mode. Using Oracle 9i to the utmost, file extraction can be done via a so-called external table. This construct, introduced in the Oracle 9i timeframe allows Warehouse Builder to treat files as relational objects, so that the extraction of flat file data can be done in parallel without first splitting up the file.

When using Warehouse Builder to extract from SAP, ABAP code is generated and moved to the SAP platform. The code is executed on the SAP platform in the SAP scheduler. In order to

achieve better scalability some extractions can be done using PL/SQL on the regular tables. Warehouse Builder allows for both implementations enabling a highly scalable extraction.

When extracting from relational systems, Warehouse Builder uses database links to access the remote system. If this is an Oracle database it is direct access. All other relational databases are accessed using a database link through gateway access (ODBC is a generic gateway) through the heterogeneous services in the database. In both cases the user would not see any difference in Warehouse Builder between all of these different sources.

Warehouse Builder adds hints to the generated code to ensure most of the data processing (for example sorting and filtering) is done on the source system. Allowing the volume of data moved across a network to be significantly reduced.

When using database links to for example other Oracle database sources, Warehouse Builder adds hints to the code to ensure most of the data processing (for example sorting and filtering) is done on the source system. This way the volume of data moved across a network is significantly reduced.

### *Transforming and loading*

Once the data is moved into an Oracle environment, Warehouse Builder leverages the transformation and load capabilities of the database. Features include pipelining extraction, complex parallel transformations and parallel loading of the warehouse.

Warehouse Builder generates PL/SQL to handle the Oracle based transformations, which provide operators to do SQL like activities. Warehouse Builder also provides out of the box transformations or user-defined transformations. The designer can create complex processes by combining all of these in graphical representations and generate either row based or set based code.

Row based code gives the user a high degree of control over what happens in the load as well as very extensive error handling capabilities. Row based processing however is typically slow. To benefit from row based processing but still achieve performance, Warehouse Builder allows the user to generate bulk processing from the same logical process. The designer can set the size of bulk arrays to be processed as one unit of work, achieving much higher throughputs with a high level of control. In most situations set based processing will however still be faster than bulk processing. Inserting into a target (with or without processing) is always faster in one SQL insert statement.

Unfortunately not all data added to a target table consists of inserts. Updates of existing records are very common. Previously these could only be handled in row-based modes when mixed with inserts in the same process. Oracle 9i solved that problem by introducing the MERGE statement in SQL. Utilizing this, inserts mixed with updates can be handled in one single SQL statement. Resulting in higher load performance, but also reduces complexity in design and code, making the maintenance and testing efforts smaller when using Warehouse Builder.

## Data storage and querying facilities

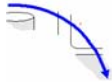
Once the data has been moved into the Oracle database, other characteristics take over and the database starts playing a different role in the architecture. Instead of moving and transforming data, it now acts as a save keeper of the data or data query engine. For this save keeper role, several database features are important - dimensions and cubes, partitioning, summary management through materialized views, indexing and query rewrite.

Data Storage

Warehouse Builder, as a design tool allows users to define and design these database structures and use them in loading and storing data. For a complete discussion on the use of these objects in data warehousing, please take a look at the Data Warehouse Guide available with the Oracle 9i documentation set.

### Parallel processing by default

One of the most important characteristics of the code generated by Warehouse Builders is its ability to use the parallel capabilities of the database. Designers can make use of these without extensive tuning activities. Warehouse Builder allows for several ways of doing parallel processing.



On the source side parallel extraction can be done, even with flat files if external tables are used. Warehouse Builder allows for parallel DML and by default enables this in the code that it generates. On the target side designers can add hints to specify the degree of parallelism.

Parallel index creation and data object creation are other features that benefits heavily from the parallel processing capabilities of the database. When creating or re-enabling an index, Warehouse Builder ensures that the processing is done in parallel if possible.

When using functions in the parallel environment, adding the parallel enabled statement (as is done in all Warehouse Builder generated ETL code) will allow all in-line functions to be performed in parallel as well allowing for full parallelism through the transformations.

### Data Quality in the engine

One of the new features in Warehouse Builder 9i is the addition of a data quality engine. As is done with the ETL engine for Warehouse Builder, the Name & Address engine is placed in the server<sup>6</sup>. This engine is an internal Oracle engine and has specially been created to handle external name and address validation libraries.



The Warehouse Builder client adds name and address cleansing capabilities to the ETL process. When generating code with this functionality in it, PL/SQL APIs are called in the database server that connects to the Name & Address server. The data is handled in this server and validated against 3<sup>rd</sup> party libraries. Since the APIs are PL/SQL they are parallel enabled and provide scalability and performance to the name & address cleansing.

Currently the available name & address libraries for Warehouse Builder are those delivered by Trillium, which allow international data cleansing and Geo Coding for the US.

### Runtime information and reporting

When the PL/SQL code runs in the database and transports data to the targets, audit information is stored in a set of database tables, the runtime repository. Warehouse Builder can store information on a record by record basis (including the values of the records) or on higher aggregation levels. The user can customize the level of auditing information without changing any logical design in the mapping. Even when the code is generated, e.g. at runtime, this auditing level can be increased or decreased by changing a runtime parameter on the job.



<sup>6</sup> This is a separately installed component in the server CD-pack required to use this functionality in Warehouse Builder

---

Warehouse Builder stores information in a runtime schema that is installed from the client. This schema holds a set of tables and packages to allow for runtime auditing. The packages and tables are generic in the sense that when Warehouse Builder generates code, calls to the packages (more specific to procedures within these packages) are made when appropriate. The audit level and the generation mode that is chosen determine the level of detail that is logged. It is even possible to switch off this auditing completely to increase the performance for clean data. The levels of auditing are:

- None; no auditing calls are made
- Statistics; showing the overall load statistics (selected, inserted, updated and number of errors)
- Error Details; showing error details including error message
- Complete; showing all records including the data of these records

Note that all of these are inclusive, meaning that details include the information logged in Statistics.

Public views are available on this runtime repository to allow designers to create a reporting environment. However to ease the task, Warehouse Builder comes with a separate viewing component on the runtime repository. This is called the Runtime Audit Browser and allows for easy browsing and drilling within the runtime repository.

---

## Conclusion

With its extendible MetaBase, Oracle storage capabilities and extensive support and usage of open standards, Warehouse Builder is a complete and reliable warehouse and ETL design environment. With the many graphical design components and web-based metadata reporting developer productivity is very high. Added to that is the increased productivity of a repository based tool with life-cycle management.

Warehouse Builder's unique capabilities to generate specially tuned code for the Oracle 9i database makes Warehouse Builder a highly scalable and reliable tool for data warehousing. Parallel enabled by default allows for fast response times even with heterogeneous sources to load from. Data quality is delivered on the same platform, allowing for usage of 3<sup>rd</sup> party vendor data files. Auditing and process information is stored in the runtime repository and easily accessible through a dedicated environment.

The combination of this strong runtime platform and an integrated metadata repository make Warehouse Builder one of the leading ETL tools in the market.

Oracle 9i Warehouse Builder  
Architecture white paper  
January 2003

Author: Jean-Pierre Dijcks

Oracle Corporation  
World Headquarters  
500 Oracle Parkway  
Redwood Shores, CA 94065  
U.S.A.

Worldwide Inquiries:  
Phone: +1.650.506.7000  
Fax: +1.650.506.7200  
[www.oracle.com](http://www.oracle.com)

Warehouse Builder inquiries  
<http://otn.oracle.com/products/warehouse/content.html>

Oracle is a registered trademark of Oracle Corporation.  
Various product and service names referenced  
herein may be trademarks of Oracle Corporation.  
All other product and service names mentioned may  
be trademarks of their respective owners.

Copyright © 2003 Oracle Corporation  
All rights reserved.