



Oracle Warehouse Builder 10g

体系结构白皮书

2004 年 2 月

目录

简介	3
概述	4
设计组件.....	4
运行时组件.....	5
设计体系结构.....	6
信息库.....	6
MetaBase.....	6
API 和 SDK 层	7
导入元数据.....	8
部署源数据和代码.....	8
客户端工具.....	9
元数据报告环境.....	10
运行时体系结构.....	11
ETL 引擎	11
数据存储和查询工具.....	12
默认并行处理.....	13
引擎中的数据质量.....	13
运行时信息和报表.....	13
结论	15

简介

Oracle Warehouse Builder 10g 是唯一一个企业级商务智能集成设计工具，可管理 **Oracle** 数据库的数据和元数据的整个生命周期。**Warehouse Builder** 利用了 **Oracle** 的商务智能特性和可伸缩性，设计人员和开发人员可使用它来为企业或部门设计、实例化、管理以及维护数据仓库。

设计和开发商务智能解决方案的过程非常复杂。选择正确的工具和供应商在降低复杂度方面起着重要作用。本白皮书详述了 **Oracle Warehouse Builder** 的体系结构，并解释了为什么对于选择 **ETL** 和设计环境用于其商务智能项目的客户来说，**Oracle Warehouse Builder** 是一个理想的工具。

所有商务智能解决方案的核心都是数据整合和集成。然而企业通常倾向于购买和使用来自不同供应商的各种工具。当他们试图跨这些完全不同的工具集成他们的元数据时，就会再次出现集成问题。**Oracle** 的价值主张是提供一个集成的解决方案，而不是为每个企业集成各个工具。因此，企业只需要完成数据集成即可。

为什么 **Oracle Warehouse Builder** 对于商务智能解决方案来说如此重要？答案可以从元数据对于企业中不同用户的重要性中得出。**ETL** 工具应该，且 **Oracle Warehouse Builder** 也确实企业的整个元数据策略中扮演了重要角色。**Oracle Warehouse Builder** 不是通过同步元数据存储来集成工具，而是自己拥有大部分查询工具信息并允许查询工具和 **OLAP** 工具使用这些元数据。**Oracle Warehouse Builder** 不是集成工具和信息，而是提供了一个现成的集成解决方案。

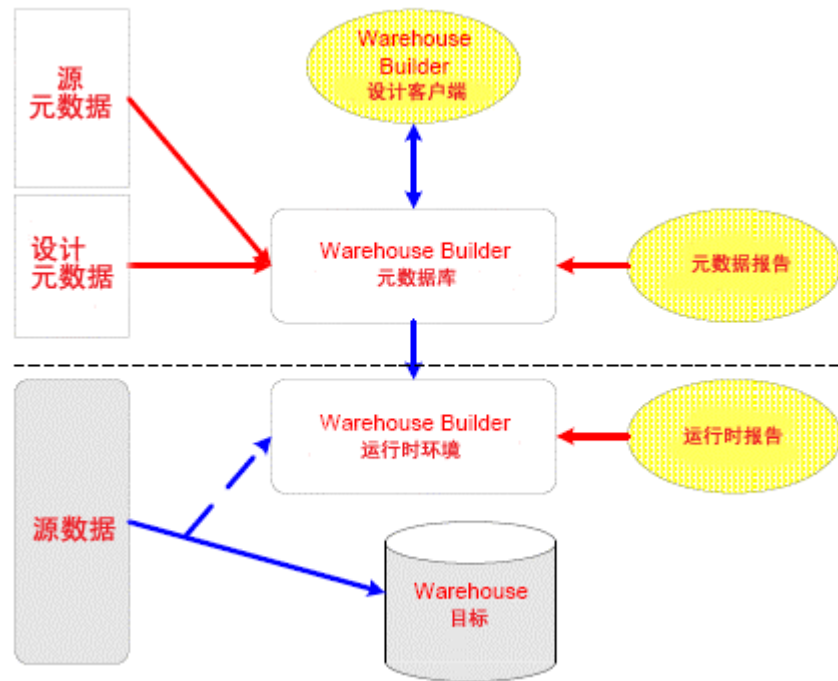
概述

Warehouse Builder 是一个基于信息库的 ETL 和数据仓库工具。其基本体系结构包括两个组件：设计时环境和运行时环境。每个组件处理系统的不同方面：设计时环境处理元数据，运行时环境处理物理数据。

元数据组件（图中上半部分）围绕着元数据信息库和设计工具。数据组件（图中下半部分）围绕着运行时环境和数据仓库。

设计组件

Warehouse Builder 设计组件包括一个存储于 **Oracle** 数据库中的高度可伸缩的元数据信息库和一组以 **Java** 或 **HTML** 语言编写的客户端设计和报表工具。可以使用这些工具来查看和操作元数据。



创建元数据是使用客户端工具在编辑器中设计对象、过程和作业的一种设计活动。以互动方式创建元数据库的方法通常用于新系统设计中。**Warehouse Builder** 支持通过客户端设计关系数据库模式、多维模式、ETL 过程和最终用户工具环境。

源系统在任何 ETL 解决方案中都扮演着重要的角色，**Warehouse Builder** 提供了集成的组件将这些将这些相关信息载入其信息库，从而代替人工创建。

这一结构的优势之一是生命周期管理，它允许基于源系统的更改来更新元数据。然后 **Warehouse Builder** 再将这些更改传播到 ETL 过程和目标系统。

为保证信息库中的元数据质量和完整性，**Warehouse Builder** 在信息库中提供了大量的验证。验证有助于保持由多用户创建的复杂系统处于一个准确且一致的状态。

为进一步方便元数据的创建和评估，**Oracle** 提供了一个基于 **Web** 的元数据报表环境。利用这一报表工具，开发人员和用户不用设计工具便能浏览和检查系统元素。该报表环境中的一个重要组件是 **Impact Analysis** 功能，使用该功能用户可以在实施更改前确定其对系统的影响。利用 **Impact Analysis** 报告可更好地控制更改并对其实施进行更好的规划。**Warehouse Builder** 还提供了一个与此相反的功能 **Data Lineage** 报表，可用于追踪数据来源所在。

运行时组件

当用户完成了逻辑层面的 ETL 系统设计后，他需要将其落实到物理数据库环境中。在完成这一步骤前，需要向逻辑设计中添加（配置）关于该数据库环境的信息。完成配置后，将生成代码。

Warehouse Builder 为 ETL 过程生成提取特定语言¹，为数据库对象生成 SQL DDL 语句。生成的代码将部署到文件系统中或数据库中。

执行 ETL 功能意味着通过例如 **Oracle Enterprise Manager** 之类的某些工具运行部署在数据库中的代码。然后，ETL 过程将源数据放入目标数据库中²。这可以是数据移动区域、操作数据存储库、数据仓库或任何其它模式。注意：**Oracle** 数据库外的代码将由各自环境执行。例如，用于从 **SAP** 系统中提取数据的 **ABAP** 代码就要在 **SAP** 环境中运行。

为报告数据负载情况，**Warehouse Builder** 生成的代码中包含有审计例程。这些例程会将关于当前负载的信息写入到 **Warehouse Builder** 运行时表中。运行代码时收集的信息包括选择、插入和更新的行数。当转换或加载数据时出错，审计例程会报告这些错误，将其写入运行时表中。为便于访问并报告该运行时信息，**Warehouse Builder** 提供了 **Runtime Audit Browser**。

相关性管理和调度功能可通过与特定的 **Oracle** 工具集成来提供。**Oracle Enterprise Manager (OEM)** 是 **Oracle** 的调度和数据库管理工具。**Warehouse Builder** 在 **OEM** 信息库中创建作业，可对这些作业连同其他数据库活动进行调度和监控。通过与 **Oracle Workflow (OWF)** 的交互，**Warehouse Builder** 用户可以为 ETL 过程（包括通知）间的相关性创建完善成熟的过程。

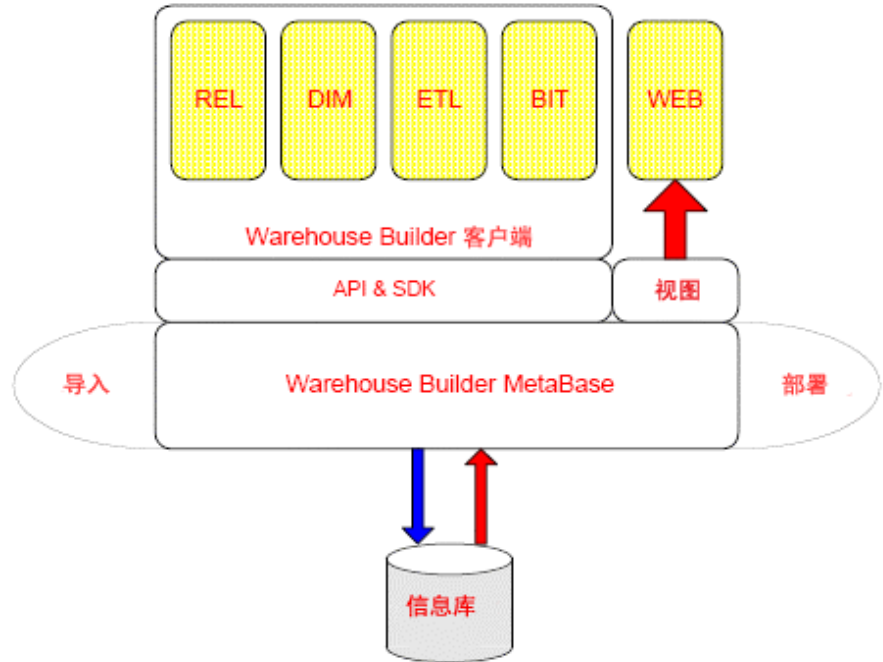
¹ 这些语言包括：用于平面文件的 **SQL*Loader** 控制文件，用于 **SAP/R3** 提取的 **ABAP** 和用于其他系统的 **PL/SQL**。

² **Warehouse Builder** 支持的目标数据库有 **Oracle 8i EE**、**Oracle 9i EE** 和 **Oracle 10g EE**（**10g** 数据库的支持需要 **Warehouse Builder 10g**）。

设计体系结构

开放性、生产率和可靠性是定义仓库设计平台中的关键词。本章将讨论 Warehouse Builder 设计体系结构的这些特性。首先概述 Warehouse Builder 体系结构的各组成模块，然后再详述这一设计体系结构的各组件。

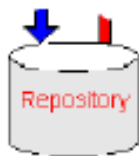
这一设计体系结构的中心概念为存储于 Oracle 数据库中的信息库。这一信息库由 Warehouse Builder MetaBase 管理，向元数据的所有产生方和使用方提供服务。体系结构图显示 MetaBase 为分发和接收元数据库的中枢。



客户端工具利用 API 提供图形编辑器，图形编辑器通过 MetaBase 服务来操作信息库中的元数据。导入和部署服务是 MetaBase 的扩展，作为 MetaBase 与外部的接口。为实现以

SQL 方式访问 MetaBase，我们提供了供元数据报告环境使用的公共视图。我们将在接下来的几个段落中详细讨论这些组件及其之间的交互。

信息库



信息库是一个细粒度层次存储数据的关系型结构。它通过存储数据（聚集在 MetaBase 中）来向用户表示对象。在本质上，信息库是一个在 Oracle 数据库中提供高度伸缩的数据存储的数据结构因为信息库运行于 Oracle 企业版数据库，所以可以保证其开放性、伸缩性和可靠性。

MetaBase



信息库的核心功能称为 Warehouse Builder MetaBase，它与数据库相分离。这一组件是元数据管理体系结构中的真正中枢，提供了开发人员所需要的大量服务。

MetaBase 实现了一个可伸缩的解决方案，可满足 Warehouse Builder 客户端设计组件的定制需求。与 MetaBase 相关的两个重要组件是导入和部署服务，我们将分别对它们进行讨论。其他主要的服务说明如下。

对象存储和检索

对象存储于数据库中。但是，为存储和检索这些对象，MetaBase 采取了一种协同和受控的机制。该机制与数据库机制相分离，以实现 MetaBase 的独立服务。

Warehouse Builder 了解 First Class Object (FCO) 的概念，First Class Object (FCO) 是由一组对象合并而成的对于用户来说有意义的对象（例如：一个表业务组件是 FCO，但表有多个 Second Class Objects (SCO)，如列和约束）。MetaBase 确保了对 FCO 的检索和存储。

多用户和锁定服务

为检索对象，并允许随后对其编辑（例如：将更改写回至信息库），需要对对象进行锁定。该对象 (FCO) 包含多个组件，因此当每次检索和编辑一个对象时，MetaBase 需要锁定所有的 SCO 以确保这些操作。因为 Warehouse Builder 是多用户的，所以这种锁定更为繁琐。第一个打开对象的用户将获取对整个 FCO 的读取/写入锁。随后的用户将只能获得读取锁。MetaBase 允许在具有写权限的用户提交更改后对视图进行刷新。MetaBase 将该提交转化为数据库提交。

框架的业务逻辑支持

Warehouse Builder 提供的所有服务都根源于 MetaBase，MetaBase 提供了一个框架，业务逻辑层位于其上。该业务逻辑层将对 MetaBase 检索和存储的对象进行解释。

业务逻辑的主要功能对于支持 Warehouse Builder 生成和验证服务极其重要。验证服务允许对元数据进行校验，以及早报告误配和错误。生成服务是编辑和提供代码的服务。

API 和 SDK 层

为实现客户端工具与 MetaBase 的连接和通信，我们提供了 SDK（软件开发工具包）和一个 API 层。为实现只读 SQL 访问，我们提供了一组公共视图。

公共 API 和脚本语言

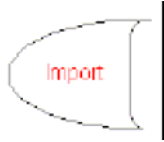
利用这一组公共 API，Warehouse Builder 用户不需使用用户界面也能对信息库进行可控式操作。实施 API 是一种分阶段项目，OWB 首个版本中发布了结构化 API。这些 API 是具有完整文档的 Java API。因为它们是公共的，所以可以保证其向后兼容性。

Warehouse Builder 基于这些 API 向用户提供了脚本语言，可以在命令行界面中使用这些脚本语言来执行元数据操作命令。脚本语言支持对信息库进行直接操作，也支持批量脚本执行。这一机制提供了一个将批量更新应用于信息库的理想方法。这一脚本语言基于 TCL 脚本语言。

公共视图

为提供到信息库的 SQL 只读式访问，Warehouse Builder 在设计和运行时信息库中都提供了公共视图。Warehouse Builder Browser 使用这些视图以安全高效的方式向最终用户报告元数据。为构建自己的报告环境，用户可以使用这些视图。这些视图具有文档且向后兼容。

导入元数据



从外部系统中检索元数据是构建数据仓库的重要一步。为从源系统中加载数据，ETL 工具至少需要了解该源系统中的一些元数据。Warehouse Builder 具备一种独特的体系结构，可以提取并核对源系统的详细信息。

从体系结构方面来看，Warehouse Builder 应用了两种不同的元数据集成解决方案。

点对点解决方案用于从特定系统提取元数据，向特定于这些系统的提取加入智能。这一解决方案的一个示例为 SAP Integrator 解决方案³。这一集成程序专门设计用于 SAP R/3 系统，通过远程功能调用 (RFC) 来访问 SAP 系统的业务对象信息库。这样 Warehouse Builder 就可以读取 SAP 系统的业务逻辑，提取那些信息并存入信息库。Warehouse Builder 使用这些信息生成特定源提取代码（在 SAP 示例中为 ABAP）。

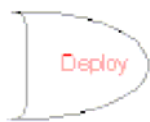
点对点元数据导入（及导出）的另一形式为通过 Metadata Loader (MDL) 导入 Warehouse Builder 自己的元数据。MDL 可以从其他 Warehouse Builder 信息库中导入元数据。这一导入功能非常灵活，甚至可以用于填补现有的元数据。

第二个解决方案为基于标准的解决方案。标准可以是内部或外部的。Warehouse Builder 的最重要的标准和源为 Oracle database catalog⁴。Warehouse Builder 支持的其他标准当然就是 OMG-CWM (Object Meta Group – Common Warehouse Method) 和 ODBC 了。

Warehouse Builder 提供了一个独有的现成功能，即可将源系统的元数据与信息库信息进行核对。这样设计人员可以通过可控且可预见的方式来添加或更改信息库元数据。Warehouse Builder 通过报表来说明对源数据的影响并可自始至终对目标的各种影响进行评估。这样一来，设计人员就可以预知影响，然后按预定时间范围传播更改。

部署源数据和代码

部署将设计落实到切实的仓库环境中。从这种意义上说，部署是连接设计体系结构和运行时体系结构的桥梁。



开发人员可使用 Warehouse Builder 将一个系统部署到不同的环境中。根据所生成代码的范围，整个模块可以部署到多个目标系统中。经过简单地更改环境信息后，同样的代码可以部署到测试和生产环境中。单个对象和整个系统都可进行部署，这提升了部署服务的灵活度和粒度。

³ 其他点对点解决方案包括平面文件集成程序、Oracle Applications integrator、Oracle Designer 和 CA ERWin 导入。后两者也可以导入到目标模块。

⁴ 透明网关通过数据库上各式连接使用这些标准，这使得 Oracle catalog 成为 Warehouse Builder 的关系型系统访问标准。

生命周期管理

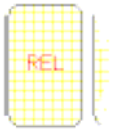
Warehouse Builder 提供的一个独特功能就是根据元数据定义的更改来管理对目标数据库的更改。Warehouse Builder 提供了更新脚本生成功能，用以代替丢弃对象的方法，这一方法会在仓库环境中导致各种问题。在应用任何这些更新脚本前，设计人员可以查看提供的 **Impact Analysis** 报告，然后选择推迟或应用更改。通过这一报告功能，用户可以根据计划方便地升级仓库，消除了大量的关机和重载操作，这使得 Warehouse Builder 的体系结构在 ETL 工具中独树一帜。

客户端工具

Warehouse Builder 包含若干设计工具，以更加完善整个仓库解决方案的设计。关系型、多维和商务智能工具设计的重要性几乎等同于 ETL 设计器中的核心功能。整个客户端软件使用 Java 编写。要了解客户端工具和功能，请参见 Warehouse Builder 使用指南或概述白皮书。以下段落将概括客户端工具的功能。

关系型设计

关系型模式设计器用于创建关系型模式。这一设计器的主要目的是让开发人员设计第三范式模式，并支持使用不同的方式在仓库环境中存储数据。这样就可以在 Warehouse Builder 中像处理维结构一样对第三范式 ODS（操作数据存储）进行建模。通过该设计器，开发人员还能在概略图中查看源代码和仓库模式，包括关系。



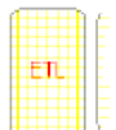
维设计

维设计器特别简化了多维结构的设计。使用它可以很方便创建一维多层、多共享层次及到事实表的连接。可以向维分配多个角色，并多次将其连接到同一事实。事实编辑器提供了一个基于向导的方法，以供用户基于信息库中的维来创建事实。Warehouse Builder 提供了事实外键的自动键和索引创建方法，以对从工具生成的事实进行星状查询转换。



ETL 设计

Warehouse Builder 的主要功能是 ETL 过程设计。在这一设计中，Warehouse Builder 将过程的物理实现和在过程的逻辑视图中设计的业务逻辑进行了分离。ETL 设计人员将首先通过指定图形中的业务逻辑来设计过程的逻辑视图。逻辑视图经配置后将转化为物理实施。指定配置将动态影响生成的代码。例如，当添加配置并声明部署环境为 Oracle 8i 数据库的新版本时，将生成不同的代码，以利用新版本中特定的 ETL 功能。不同的配置可以用于不同的环境（开发和生产）。



商务智能设计

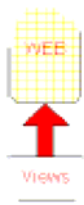
利用商务智能设计器，Warehouse Builder 用户可以为 Oracle Discoverer End User Layer (EUL) 创建元数据，并在 Warehouse Builder 中对其进行填充。利用它可以指定 EUL 对象的属性，并能为这些对象从 Warehouse Builder 信息库中的对象中导出元数据。例如一个维对象可以成为一个文件夹对象，并继承层次作为项目类。维中的的所有属性将变为文件夹中的项目。



这些丰富的客户端功能将 **Warehouse Builder** 打造成为一个灵活的设计和 ETL 工具，为 Oracle 环境生成优化的代码并提供了紧密的集成。

元数据报告环境

为了便于用户访问 **Warehouse Builder** 信息库中的重要信息，Oracle 另外采用了一种机制，



使得用户可以在综合报表中查看元数据。**Oracle Portal** 作为提供平台在一个用户友好且安全的环境中为报表提供了浏览功能。

Warehouse Builder 元数据报表基于浏览器，并继承了 **Portal** 的安全性。报表将分为多个栏目，包括浏览器环境管理和实际的元数据浏览栏目。

所有这些报表都基于公共视图运行，这意味可以使用其他工具访问浏览器报表上显示的所有信息。可以方便地对报表进行扩展，在其中加入定制信息，结合 **Discoverer** 栏目以创建一个面向管理员和最终用户的综合仓库门户。

如果您不想使用 **Oracle Portal** 的安全且可伸缩的框架，可以使用 **HTTP** 服务器在信息库数据库上以独立模式运行元数据报表。这样可以获得所有的报表功能，但不再绑定到 **Oracle Portal**。

在客户端计算机中无需安装任何 **Warehouse Builder** 软件即可从浏览器中启动所有这些报表。也可从客户端工具中启动。通过结合来自图形线性和影响分析报告的信息，仓库设计人员可以快速地更改仓库设计，以适应不断变化的环境。

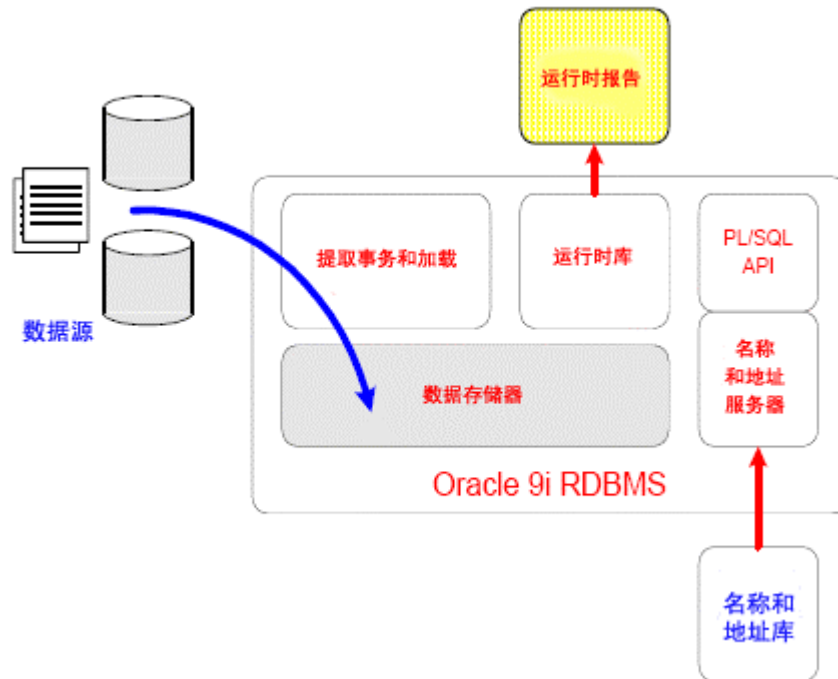
运行时体系结构

仓库系统的运行时体系结构和其设计时环境同样重要。可靠性、可伸缩性和开放性是定义一个理想的仓库运行时环境的关键。

体系结构图显示所有 Warehouse Builder 特定组件运行的基础平台是 Oracle 数据库。

这是与市场上众多 ETL 工具的主要不同点之一。Warehouse Builder 本身没有运行 ETL 作业的外部引擎。作业（生成的代码）运行于 Oracle 数据库引擎中。这样即是将 ETL 过程送往数据，而不是相反。这种与数据库引擎紧密联系的另一优势是继承了该平台的可伸缩性和可靠性。利用 Oracle 数据库中内建的 ETL 特性，

Warehouse Builder 可以向企业提供针对在其 Oracle 数据库平台上运行而进行优化的代码，从而减少了进行更多的调整和培训的需要。



ETL 引擎



Warehouse Builder 生成的代码利用了 Oracle 数据库的许多特性。数据库在运行时体系结构中扮演了两个角色。本段将着重讨论 ETL 引擎的功能和及其在 Warehouse Builder 代码中的应用。下一段将阐述另一个角色 — 数据存储和检索引擎。

提取数据

典型的 ETL 过程包括三个阶段。第一阶段是从源系统中提取数据。Warehouse Builder 的体系结构支持使用不同的方法从不同平台将数据提取到 Oracle，在 Oracle 目标平台中运行尽可能多的代码。

Warehouse Builder 调用几个实用工具从源平台中提取数据。对于文件，默认方法为使用 SQL*Loader — Oracle 数据库提供的批量加载工具。利用 SQL*Loader 可以从平面文件中进行快速提取，提取模式可以为直接路径模式或更为可控的常规模式。为充分发掘数据库的功能，可以通过所谓的外部表来进行文件提取。利用这一在 Oracle 9i 时期推出的构造，Warehouse Builder 可将文件视为关系对象，因此不必先分离文件就能以并行方式提取平面文件数据。

当使用 Warehouse Builder 从 SAP 中提取数据时，将生成 ABAP 代码并将其送往 SAP 平台。代码在 SAP 平台上的 SAP 调度程序中运行。为获得更好的可伸缩性，可以在常规表上使用 PL/SQL 来完成一些提取操作。Warehouse Builder 对于两种实施均支持，实现了高度可伸缩的提取。

当对关系型系统执行提取操作时，Warehouse Builder 使用数据库链接来访问远程系统。如果是 Oracle 数据库，则可以直接访问。其他关系数据库的访问则需要使用数据库链接通过网关（ODBC 是通用网关）、利用数据库中不同的服务来完成。两种情形中，使用 Warehouse Builder 的用户将察觉不出这些数据源之间的任何区别。

Warehouse Builder 向生成的代码加入了提示，以确保在源系统中完成大部分的数据处理过程（如排序和筛选）。从而大幅度降低网络中传输的数据量。

例如，当使用连接到其他 Oracle 数据库源的数据库链接时，Warehouse Builder 会向代码中加入提示以确保在源系统中完成数据处理的大部分过程（如排序和筛选）。这样就大幅度降低了网络中传输的数据量。

转换和加载

当数据转移到 Oracle 环境后，Warehouse Builder 将利用数据库的转换和加载功能。这些功能包括管道提取，复杂并行转换和仓库并行加载。

Warehouse Builder 生成 PL/SQL 来处理基于 Oracle 的转换，这些转换提供了操作符来完成 SQL 式的活动。Warehouse Builder 还提供了现成的转换和用户定义的转换。设计人员可以在图形表示中结合使用所有这些方法，以创建复杂的过程，生成基于行或集的代码。

基于行的代码给予用户高度控制加载过程的能力和非常强大的错误处理能力。但基于行的处理速度非常慢。为获益于基于行的处理并同时获得高性能，Warehouse Builder 允许用户从同一逻辑过程中生成批量处理。设计人员可以设定一次作业要处理的批量数组的大小，从而对更高的数据吞吐量进行良好控制。但在大多数情况下，基于集的处理仍然快于批量处理。使用一个 SQL 插入语句插入目标（处理或不处理）的速度永远更快一点。

但不是所有加入目标的数据都是插入操作。对现有记录进行更新的操作非常常见。以前只能在基于行的模式中完成这些操作，在同一过程中还要混和进行插入操作。为解决这一问题，Oracle 在 SQL 中引入了 MERGE 语句。使用这一语句，可以在一个 SQL 语句中处理插入和更新操作。这进一步提升了加载性能，同时降低了设计和代码中的复杂度，从而减少了使用 Warehouse Builder 时的维护和测试工作。

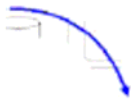
数据存储和查询工具

Data Storage

当将数据加载到 Oracle 数据库后，情况有所变化，数据库开始在体系结构中扮演不同的角色。它将不再移动和转换数据，而是负责数据或数据查询引擎的保存。对于这一角色来说，下列数据库特性非常重要——维、多维数据集、分区，摘要管理（通过物化视图）、索引和查询重写。

用户可以使用 **Warehouse Builder** 这一设计工具来定义和设计这些数据库结构，再将它们用于数据加载和存储过程中。要全面了解这些对象在数据仓库中的应用，请参见 **Oracle** 数据库文档集中的《数据仓库指南》。

默认并行处理



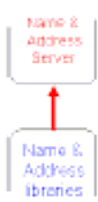
Warehouse Builder 生成的代码的重要特点之一是它可以利用数据库的并行处理能力。设计人员不需进行繁琐的调整就可利用这些功能。**Warehouse Builder** 支持多种并行处理方式。

在源端可以进行并行提取，即便源是平面文件（假如使用了外部表）。**Warehouse Builder** 支持并行 **DML**，且默认在其生成代码中启用这一特性。在目标端设计人员可以加入提示以指定并行度。

其他极大地获益于数据库的并行处理能力的功能为并行索引创建和数据对象创建。当创建或重新启用索引时，如果可能，**Warehouse Builder** 将保证以并行方式进行处理。

当在并行环境中使用函数时，加入启用了并行特性的语句后（如同 **Warehouse Builder** 生成的 **ETL** 代码一样），将以并行方式执行所有的内联函数并在转换过程中实现完全的并行。

引擎中的数据质量



Warehouse Builder 中的一个新增功能是数据质量引擎。如同 **Warehouse Builder** 的 **ETL** 引擎，名称和地址引擎将置于服务器中⁵。该引擎是 **Oracle** 的一个内部引擎，专门用于处理外部名称和地址验证库。

Warehouse Builder 客户端向 **ETL** 过程添加名称和地址清理功能。当生成带有这一功能的代码时，将在连接到名称和地址服务器的数据库服务器中调用 **PL/SQL** **API**。数据将在这一服务器中进行处理，并对比第三方库进行验证。因为 **API** 是 **PL/SQL**，所以他们具备并行能力，同时提升了名称和地址清理功能的伸缩性和性能。

Warehouse Builder 目前的可用名称和地址库由 **Trillium** 和 **First Logic** 提供。其他供应商使用适配器技术也可将他们的库插入 **Warehouse Builder** 基础架构。

Warehouse Builder 9.2 和 **10g** 的映射编辑器中内嵌有匹配/合并功能，可以进行高级的数据复制。

运行时信息和报表



当 **PL/SQL** 代码运行在数据库上并向目标传输数据，审计信息存储于一组数据库表中，即运行时信息库。**Warehouse Builder** 可以基于每个记录存储信息（包括记录的值），也可以基于更高级别的聚合来存储信息。用户不需要更改映射中的任何逻辑设计就可定制审计信息的级别。比如，即使代码生成于运行时，也可以在作业中通过调整运行时参数来提高或降低审计级别。

⁵ 这是一个独立安装的组件，位于在 **Warehouse Builder** 中使用此功能所必备的服务器 **CD** 包中。

Warehouse Builder 将信息存储于从客户端安装的运行时模式中。该模式存储了一组表和包以支持运行时审计。当 **Warehouse Builder** 生成代码时，将在合适的情况下调用包（更确切的说，调用这些包中的过程），从这一意义上来说，包和表是同一类的。所选择的审计级别和生成模式决定了所记录的详细信息的级别。甚至可以完全关闭审计功能以提升数据清理的性能。审计级别有：

- **None:** 不进行审计调用
- **Statistics:** 显示总体加载统计信息（已选、已插入、已更新和错误数）
- **Error Details:** 显示错误详细信息，包括错误消息
- **Complete:** 显示全部记录，包括这些记录的数据

注意，所有这些都是包含的，即详细信息包括记录在统计中的信息。

针对该运行时信息库还提供了公共视图，设计人员可使用它来创建报表环境。但为了简化这一任务，**Warehouse Builder** 提供了一个独立的运行时信息库查看组件。这一组件名称为 **Runtime Audit Browser**，使用它可以方便地查看运行时信息库。

结论

Warehouse Builder 包含可扩展的 **MetaBase**、**Oracle** 存储功能和广泛的开放标准支持和使用，是一个完整且可靠的仓库和 **ETL** 设计环境。其丰富的图形设计组件和基于 **Web** 的元数据报表保证了开发人员的高生产率。此外还有基于信息库的工具和生命周期管理。

Warehouse Builder 具备生成专为 **Oracle** 数据库而优化的代码的特有功能，这使其成为极其适用于数据仓库的高度伸缩且可靠的工具。默认支持的并行性将实现快速响应时间，即使是从不同源加载数据。数据质量在同一平台上提供，从而支持使用第三方面供应商数据文件。审计和过程信息存储于运行时信息库中，可通过专用环境轻松访问。

强健的运行时平台加上集成元数据信息库使得 **Warehouse Builder** 成为市场上领先的 **ETL** 工具之一。

Oracle Warehouse Builder
体系结构白皮书
2004 年 2 月

作者: Jean-Pierre Dijcks

Oracle Corporation
全球总部
500 Oracle Parkway
Redwood Shores, CA 94065
U. S. A.

全球咨询热线:
电话: +1.650.506.7000
传真: +1.650.506.7200
www.oracle.com

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

Oracle 是甲骨文公司的注册商标。
本文中提及的各种产品和服务的名称可能是甲骨文公司的商标。
其他所有提及的产品和服务名称可能是各自所有者的商标。

版权 © 2004 Oracle 公司
保留所有权利。