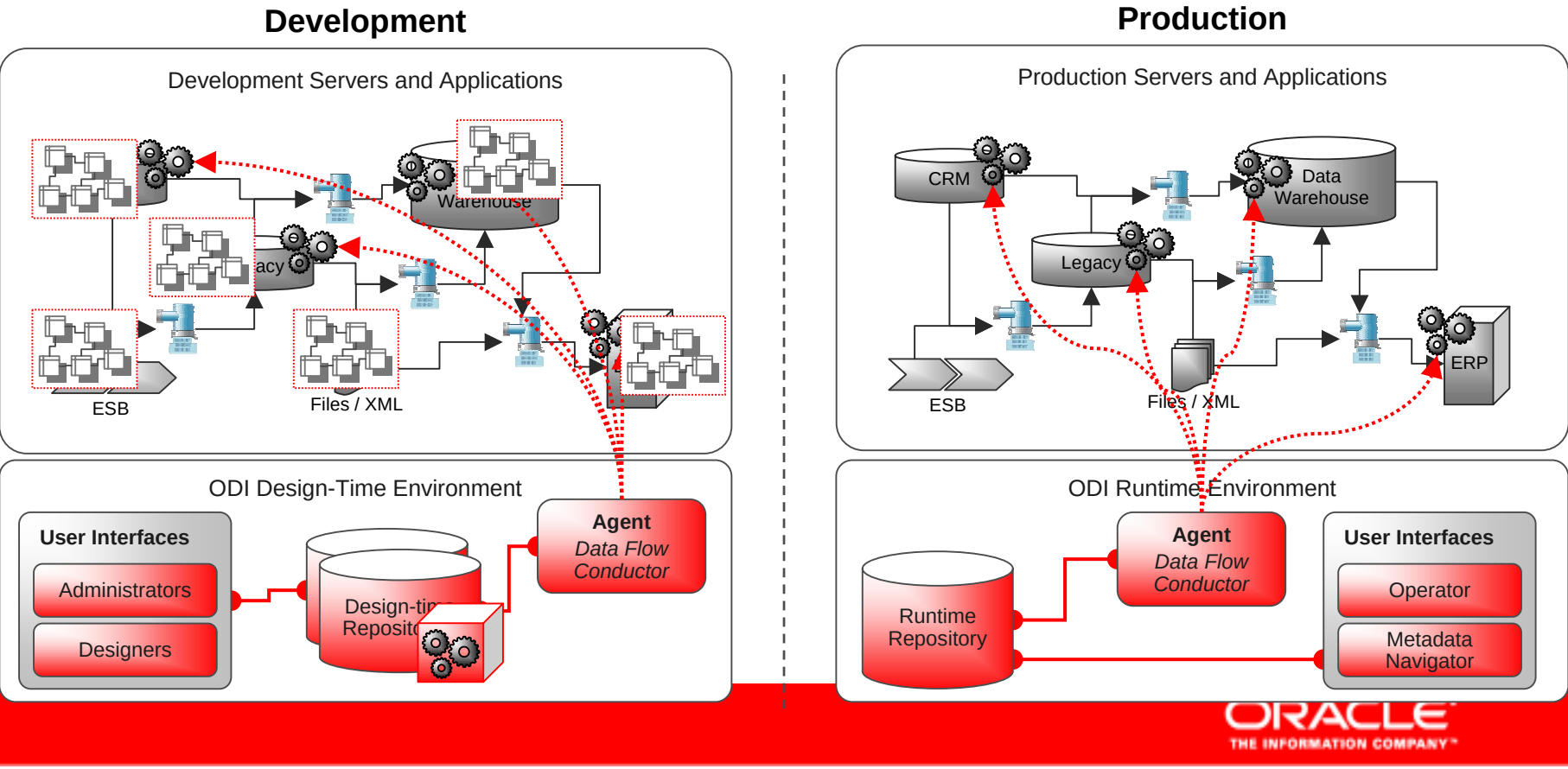
ORACLE

Oracle Data Integrator 技术介绍

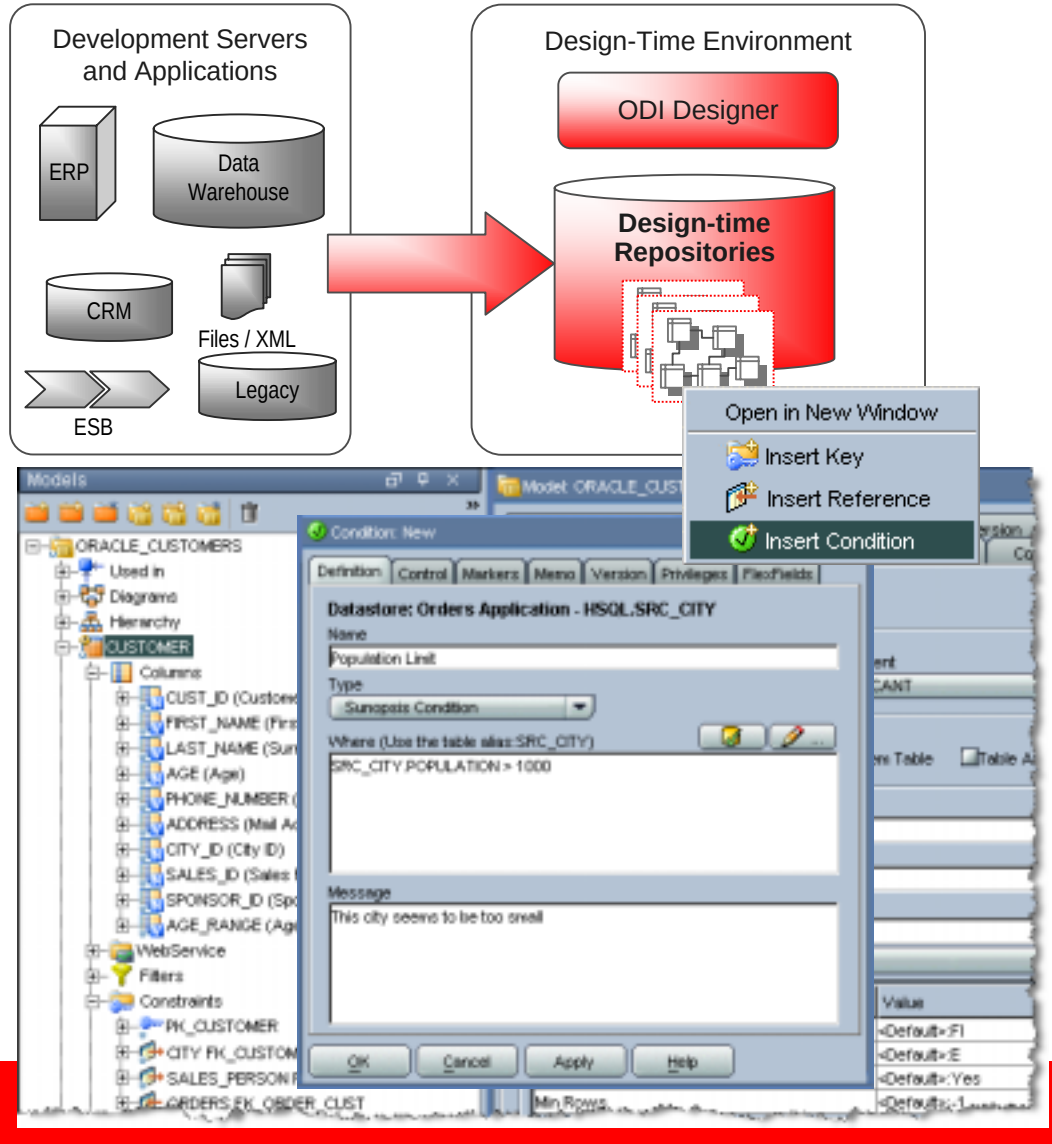
Oracle Data Integrator
6 个操作步骤

ODI 的6 个操作步骤

1. 导入/完善元数据
2. 设计转换
3. 定义数据流
4. 生成/部署数据流
5. 运行监控
6. 影响分析/数据关联



1 导入/完善元数据



1.导入元数据

- 自动的
- 客户化的
- 支持40+ 技术平台

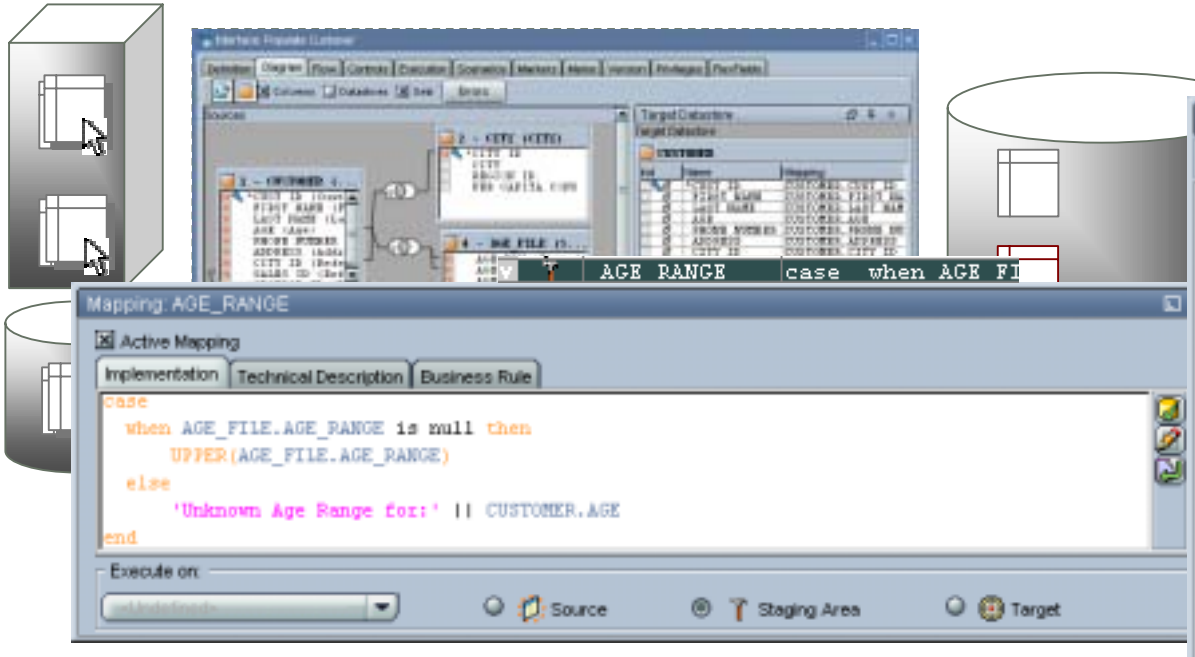
2.完善元数据

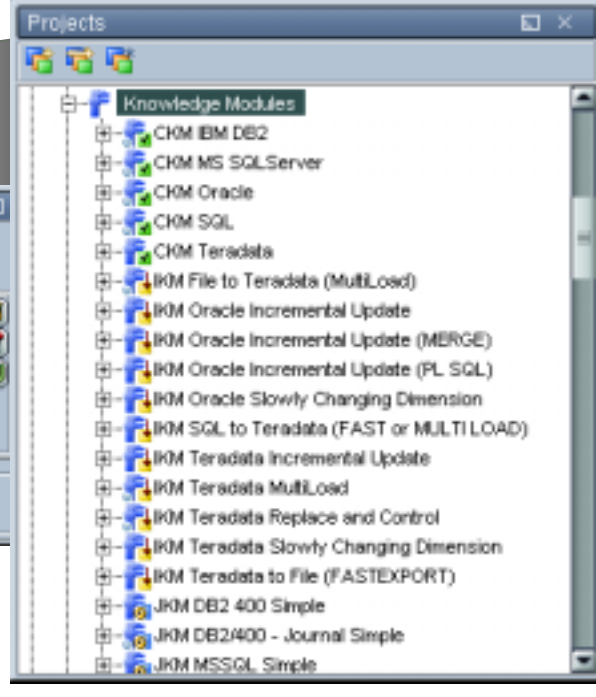
- 文档化
- 定义数据完整性规则
- 跨技术平台的参照完整性

设计转换

1 定义你需要什么

3 自动生成数据流



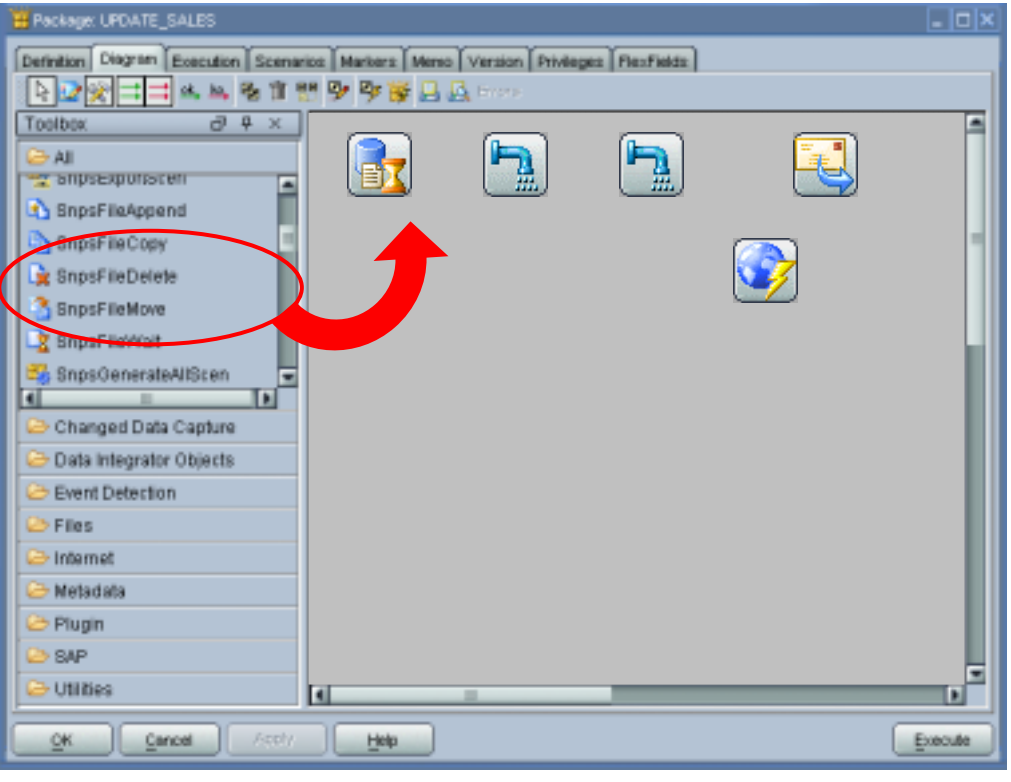


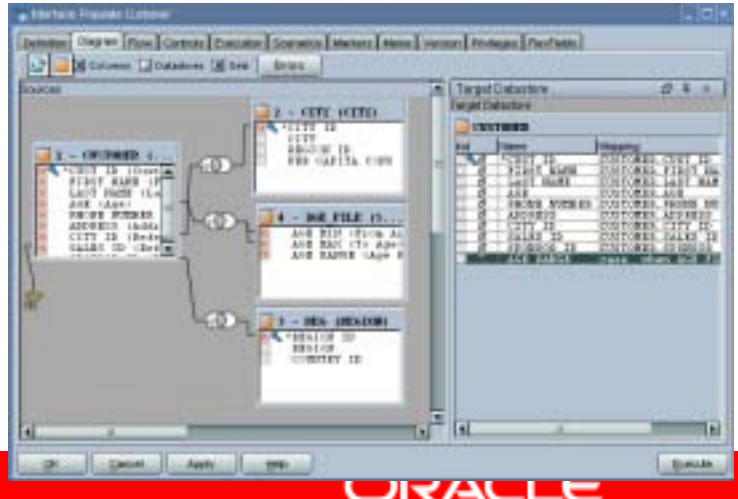
2 定义如何做: 选择模板

批量加载 • 变化数据捕获 • 增量更新 • 缓慢变化维

3 定义数据流

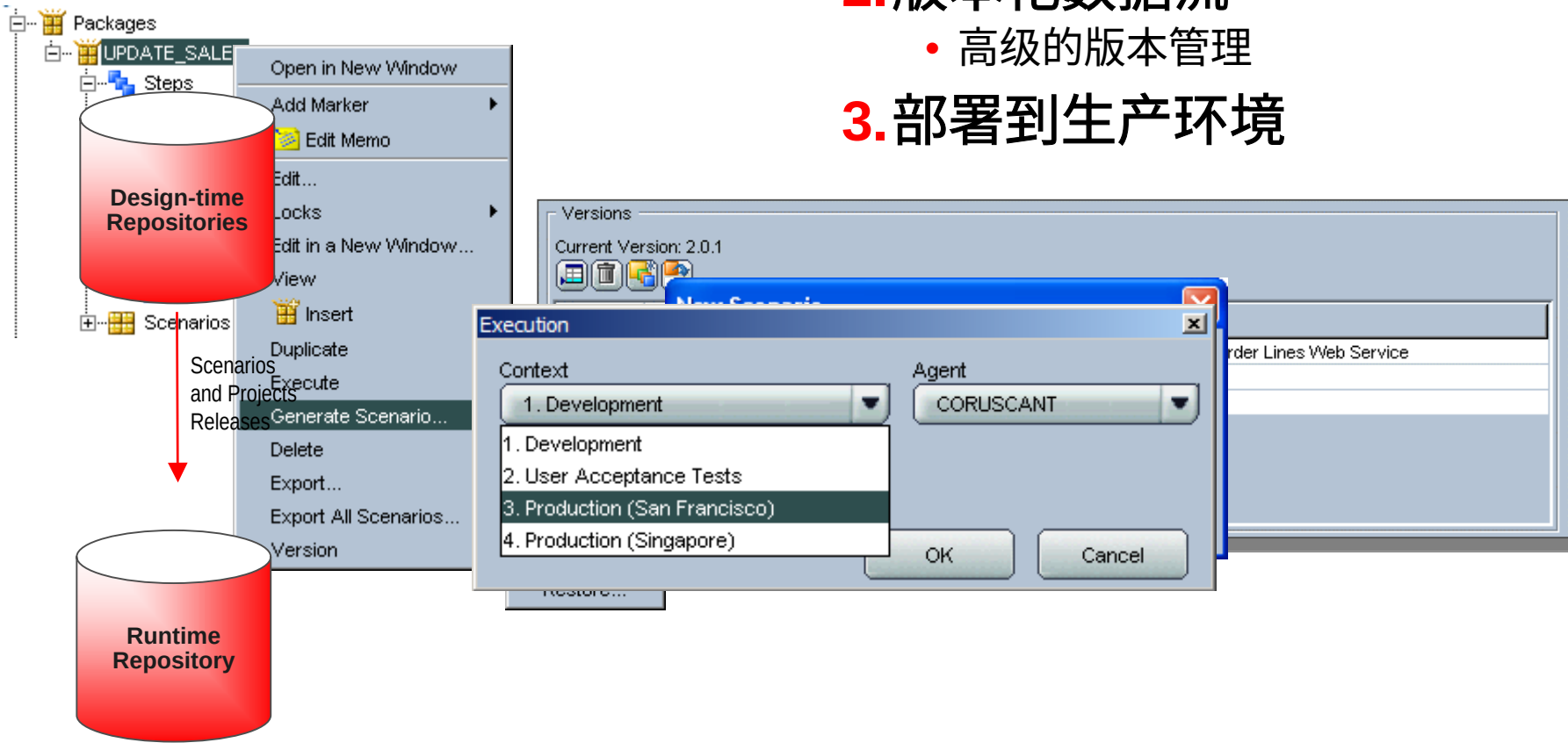
1. 确定转换流程
2. 使用ODI 的工具
   - 数据质量处理
   - 文件/档案管理
   - 发送/接收Email
   - 调用Web Services
   - 事件捕获
   - 创建自己的工具
3. 用来控制结构
   - 循环
   - 条件
   - 出错捕获



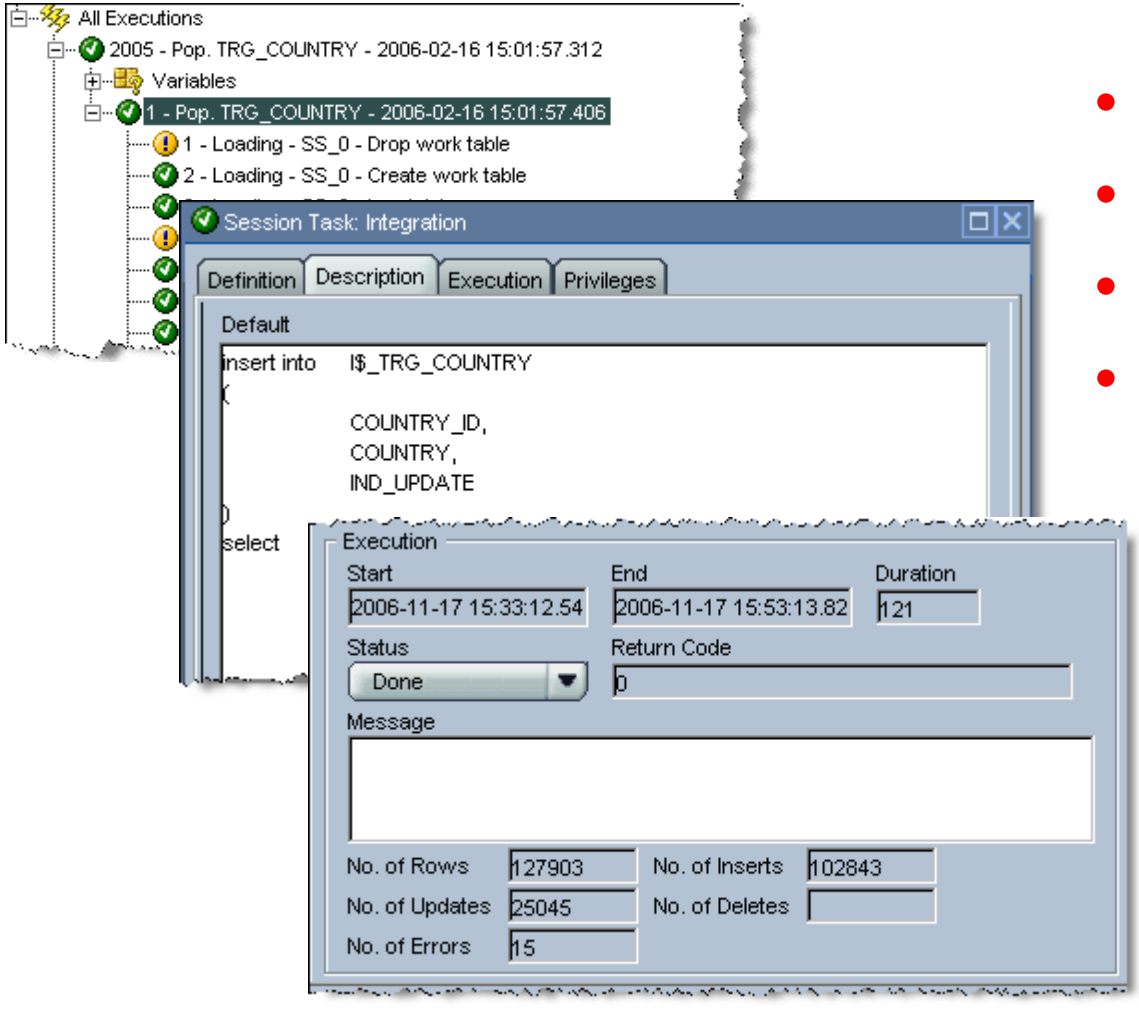


4 生成/部署数据流

1. 创建Scenarios
   - 编译运行的数据流
2. 版本化数据流
   - 高级的版本管理
3. 部署到生产环境

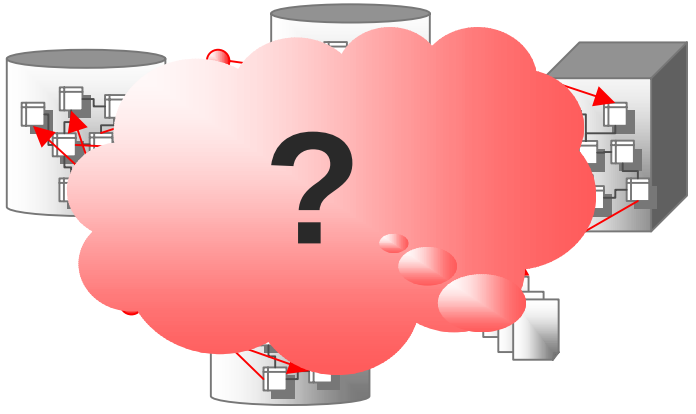


5 运行监控

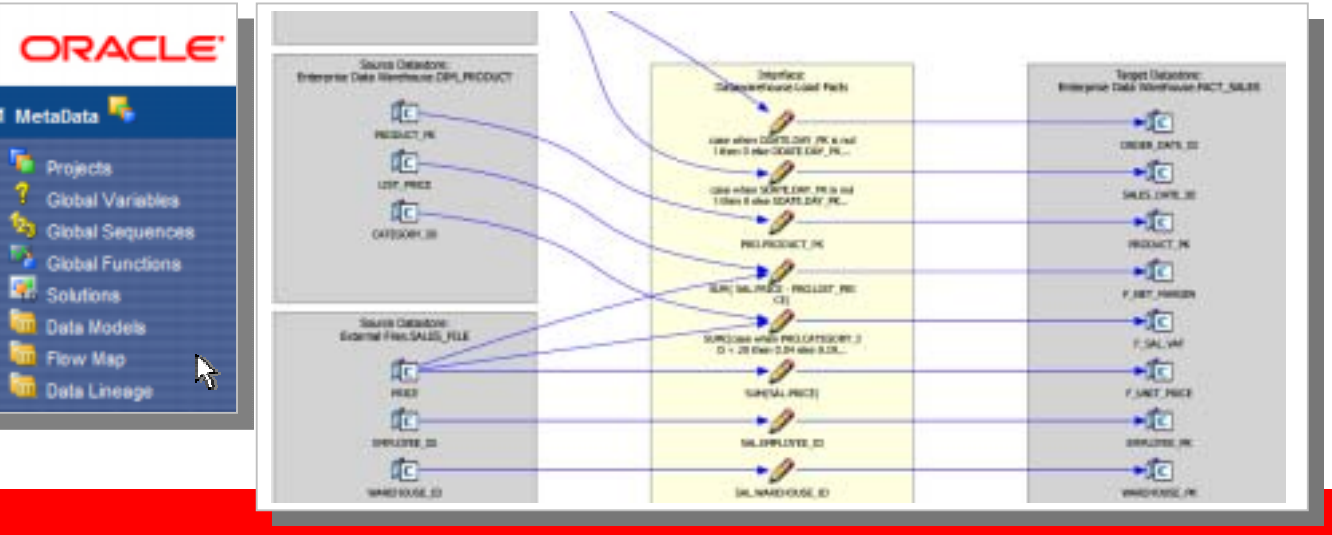


- 实时监控运行进程
- 检查执行脚本
- 详细的运行统计信息
- 重新启动失败的进程

6 影响分析/数据关联

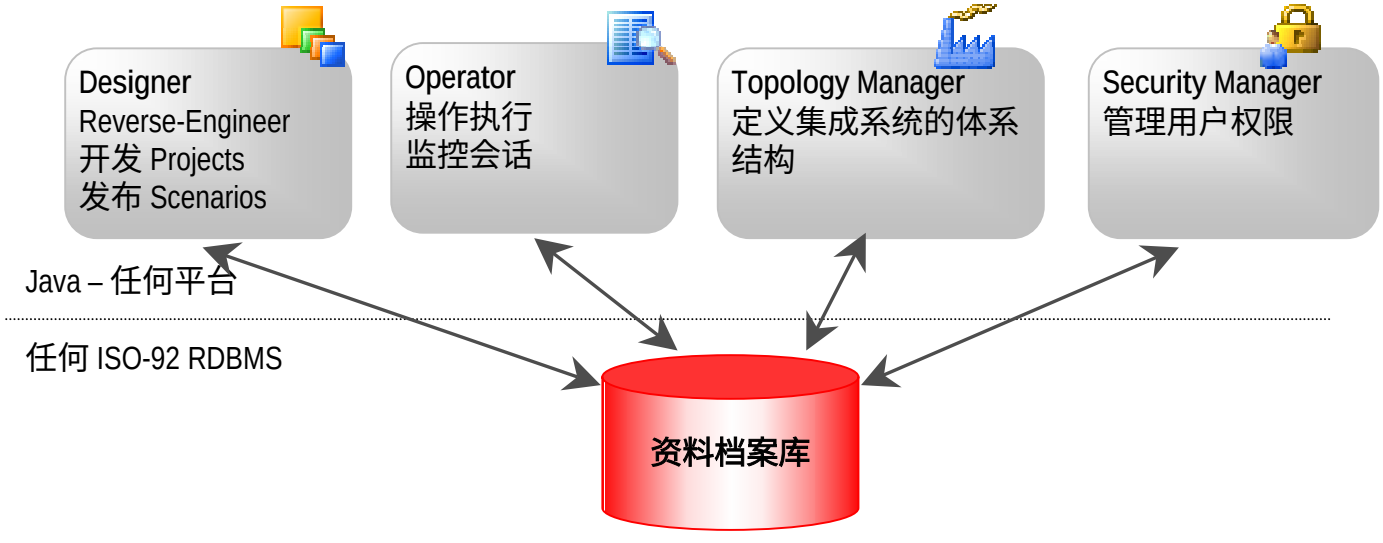


- 在复杂的环境中，维护大量的数据流
- 基于Web的端到端的数据关联分析
  1. 了解你的数据流
  2. 跟踪数据路径
  3. 钻取到转换中

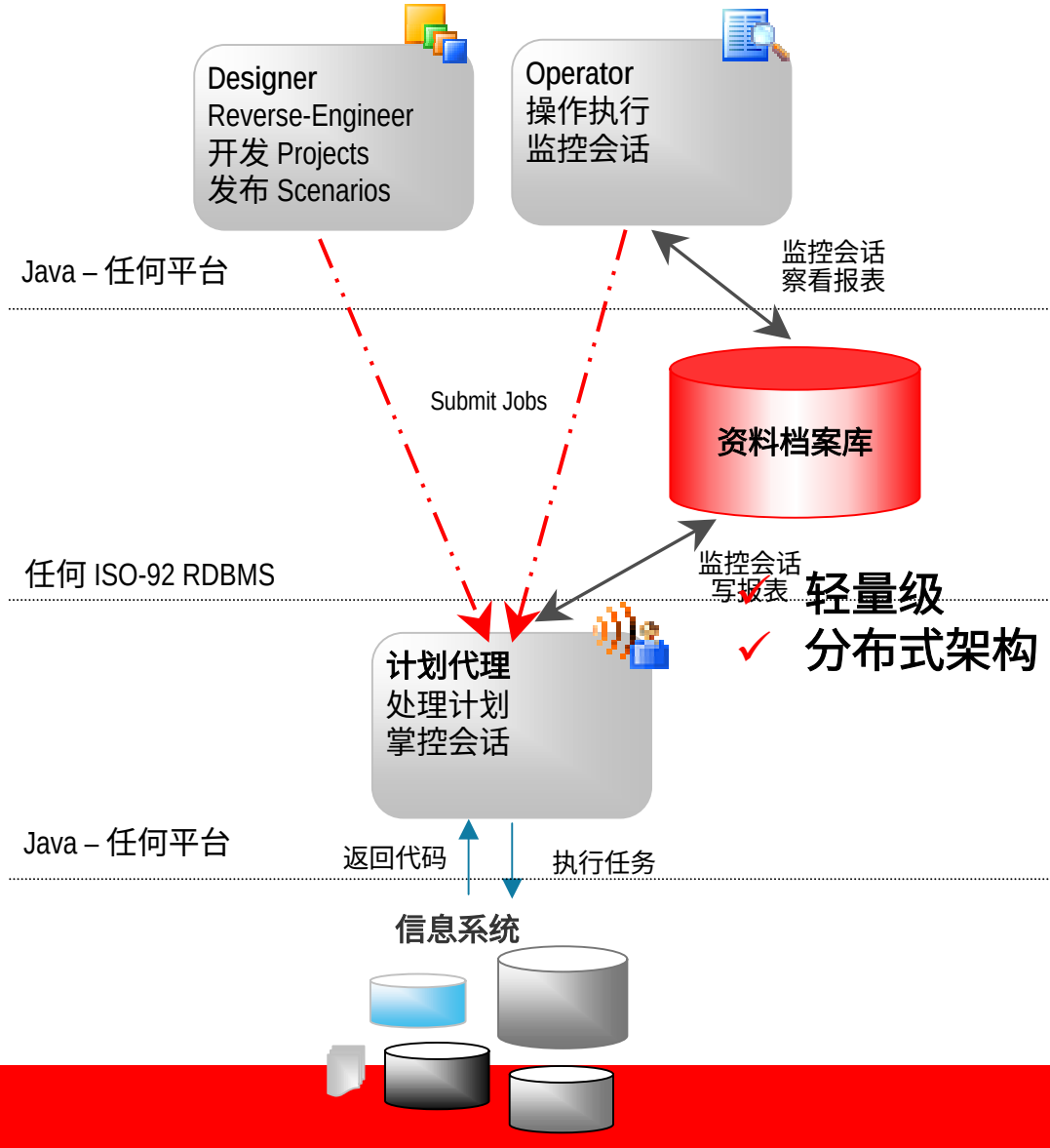


Components

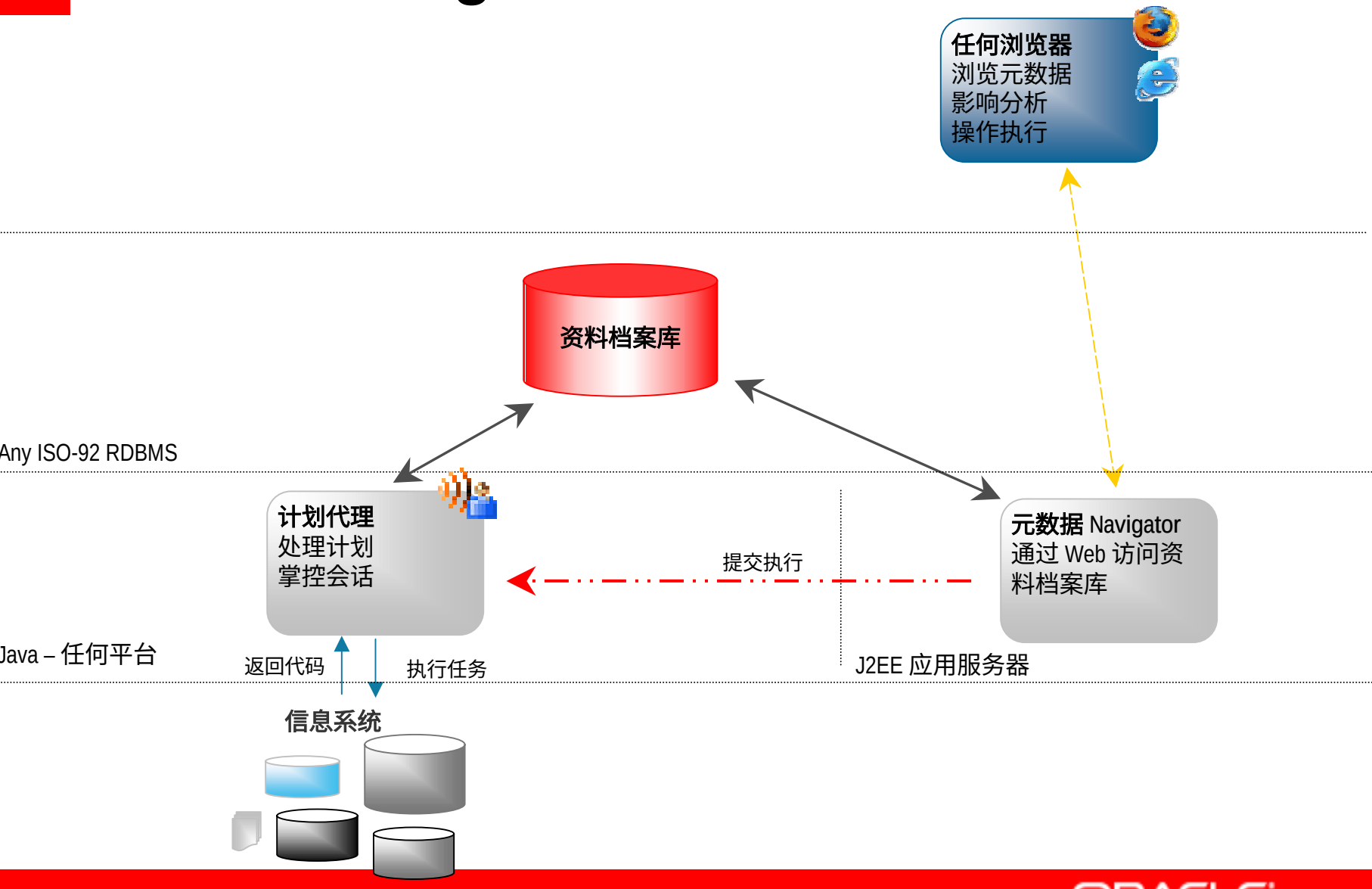
GUI 模块


Designer
Reverse-Engineer
开发 Projects
发布 Scenarios
Operator
操作执行
监控会话
Topology Manager
定义集成系统的体系结构
Security Manager
管理用户权限
Java – 任何平台
任何 ISO-92 RDBMS
资料档案库

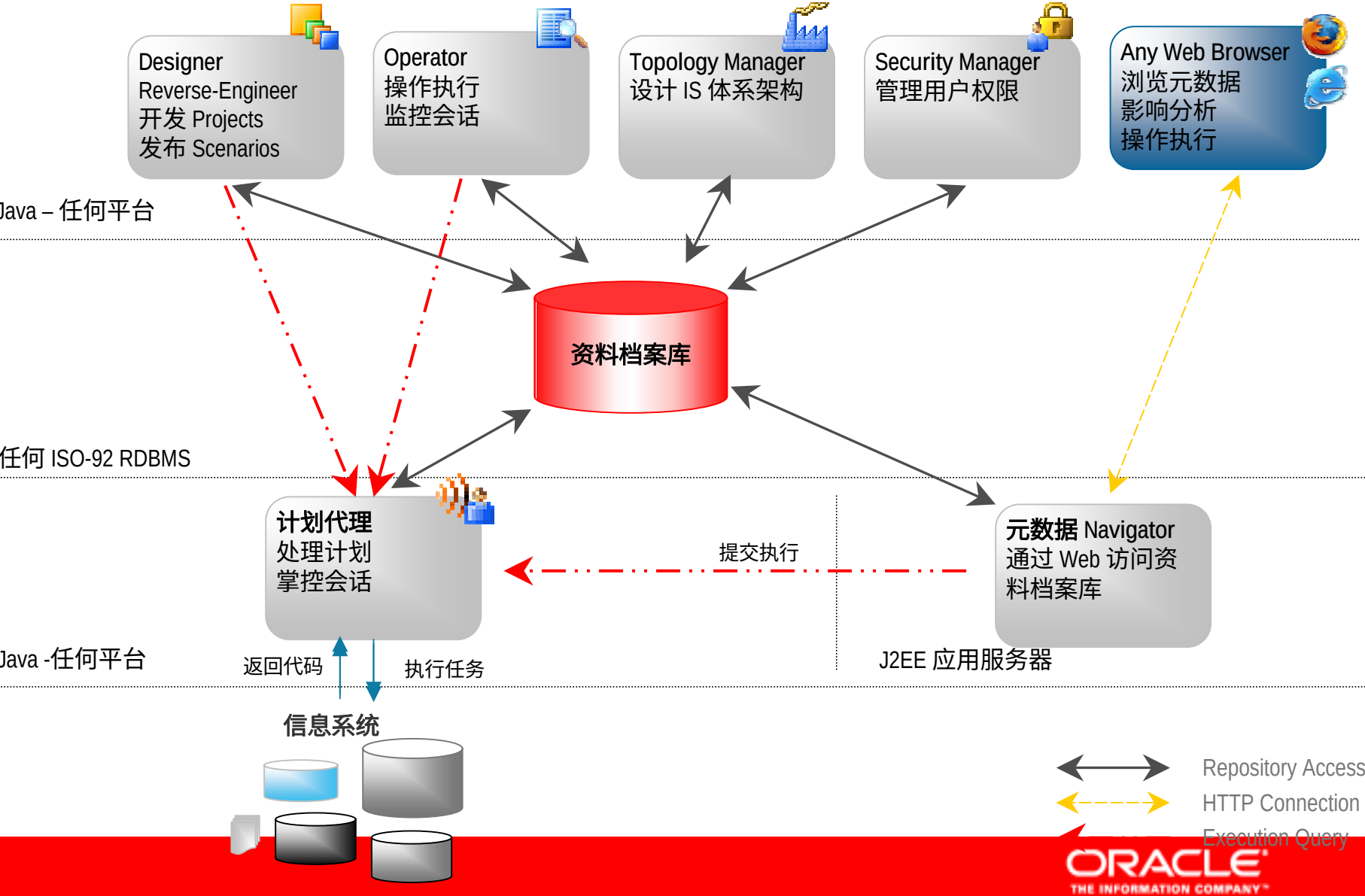
运行时组件


Designer
Reverse-Engineer
开发 Projects
发布 Scenarios
Operator
操作执行
监控会话
Java – 任何平台
监控会话
察看报表
Submit Jobs
资料档案库
任何 ISO-92 RDBMS
监控会话
写报表
轻量级
分布式架构
计划代理
处理计划
掌控会话
Java – 任何平台
返回代码
执行任务
信息系统

元数据 Navigator

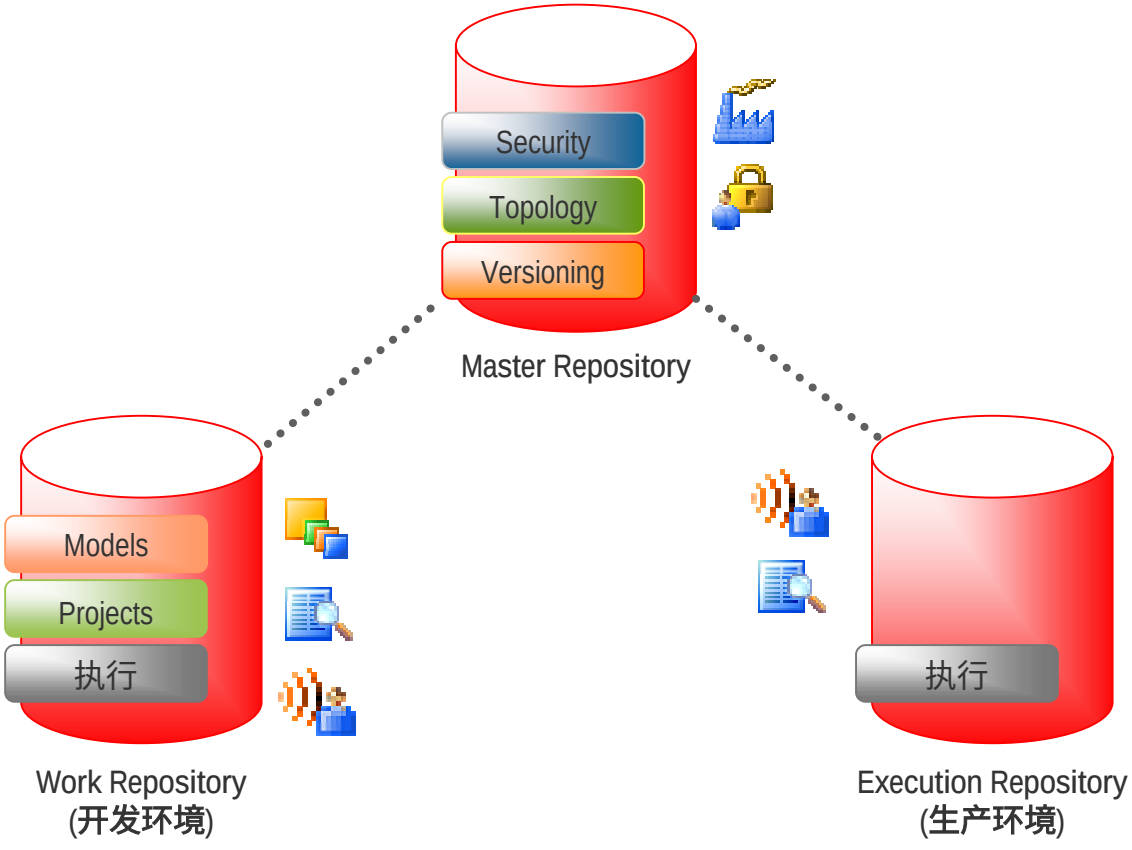


组件: 一个全局的视图


Designer
Reverse-Engineer
开发 Projects
发布 Scenarios
Operator
操作执行
监控会话
Topology Manager
设计 IS 体系架构
Security Manager
管理用户权限
Any Web Browser
浏览元数据
影响分析
操作执行
Java – 任何平台
资料档案库
任何 ISO-92 RDBMS
计划代理
处理计划
掌控会话
提交执行
元数据 Navigator
通过 Web 访问资料档案库
Java -任何平台
返回代码
执行任务
J2EE 应用服务器
信息系统
Repository Access
HTTP Connection
Execution Query

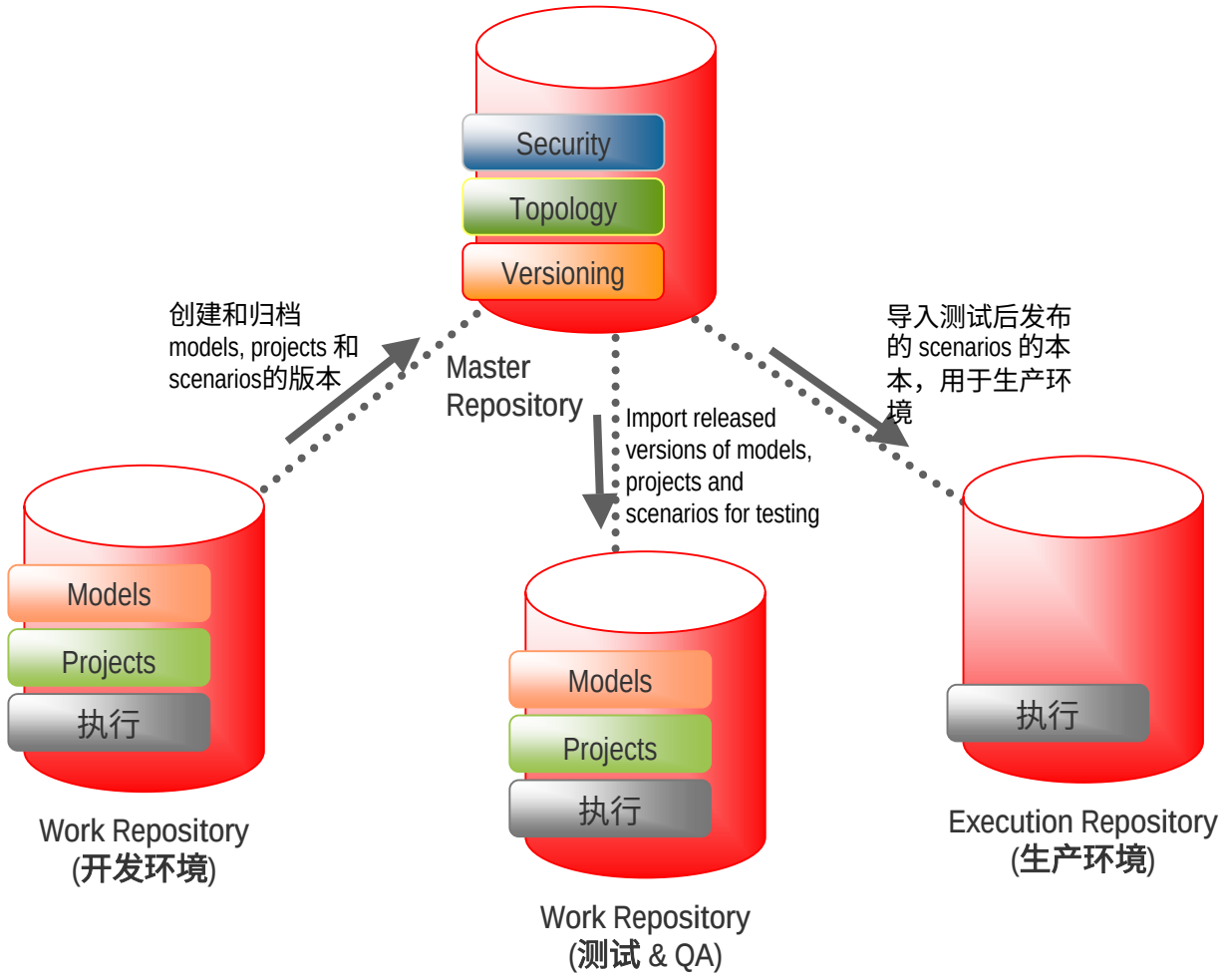
ODI 资料档案库

Master 和 Work 资料档案库



- ✓ **Two type of Repositories: Master and Work**
- ✓ **Work Repositories are always attached to a Master Repository**

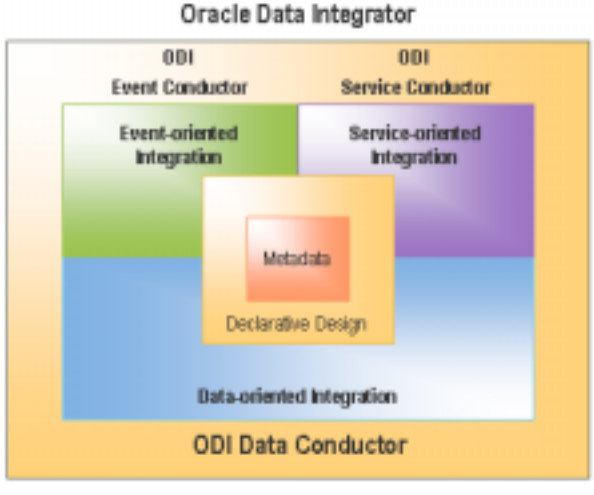
一个资料档案库配置的例子



开发 – 测试 – 生产 生命周期


EXERCISE

ODI 安装及配置


Oracle Data Integrator
ODI
Event Conductor
ODI
Service Conductor
Event-oriented Integration
Service-oriented Integration
Metadata
Declarative Design
Data-oriented Integration
ODI Data Conductor

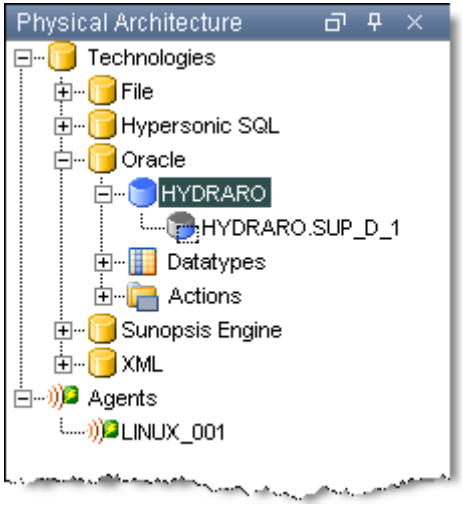
安装步骤

- JDK1.5
- Tomcat安装
- ODI10安装
- Java Driver文件拷贝
- ODI主资料库和工作资料库配置
- ODI Schedule Agent配置
- Metadata Navigator安装
- Metadata Navigator配置

ODI 连通性

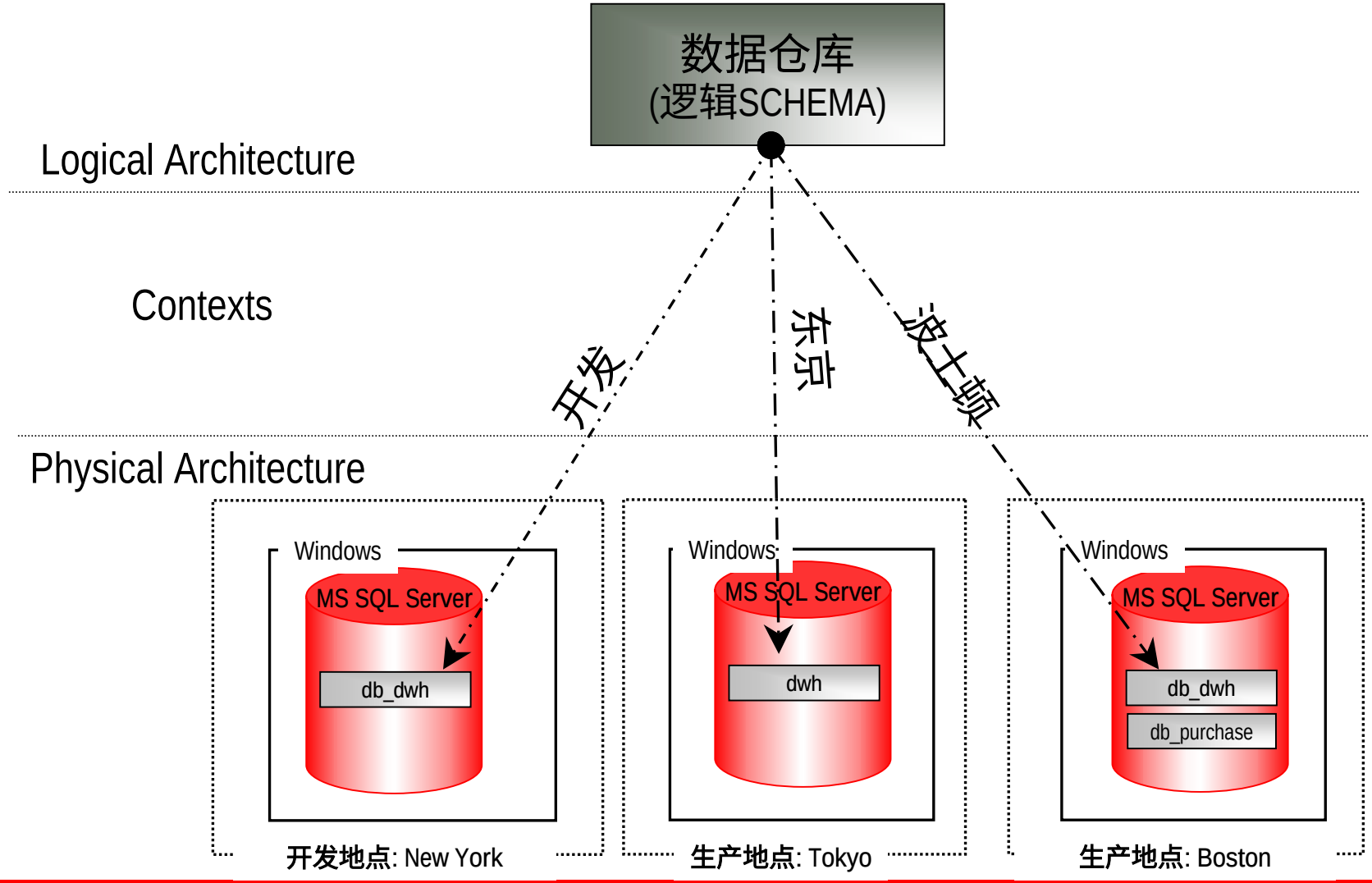
目标

完成这一部分后, 你将知道:

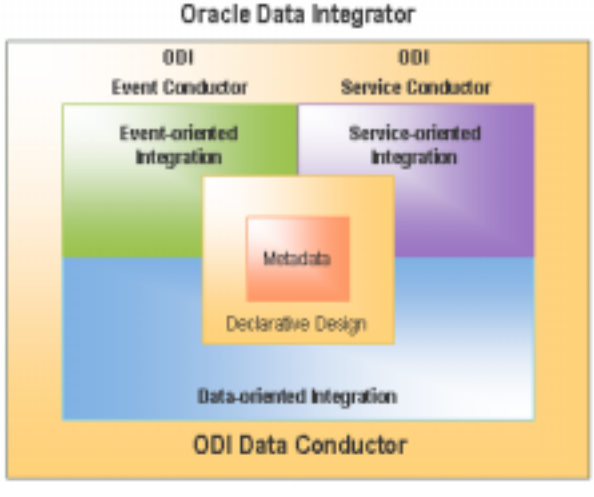


- Physical **architecture:** 定义物理体系结构 **(**运行实的连接参数**)**
- Logical **architecture:** 让我们能够连接到任何环境的抽象层
- 定义 contexts 用来表示各个环境

映射 逻辑的 和 物理的资源



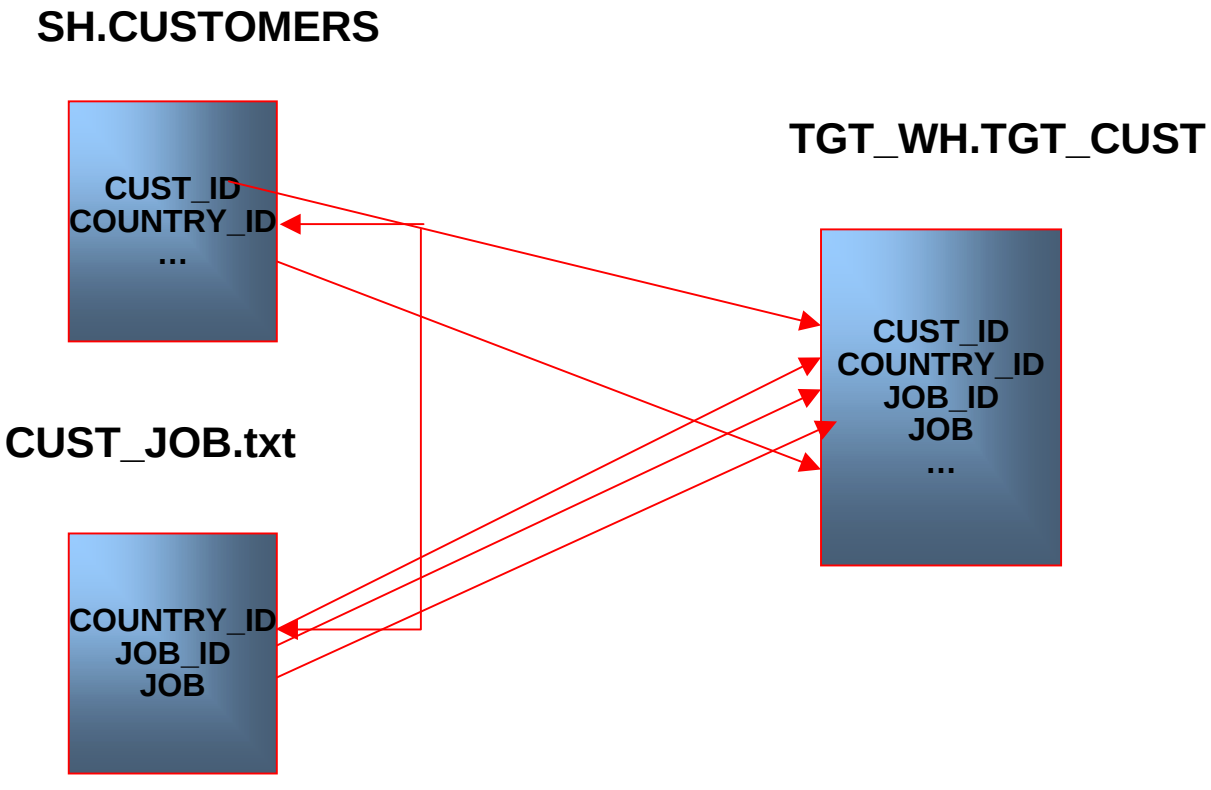
练习1： Topology



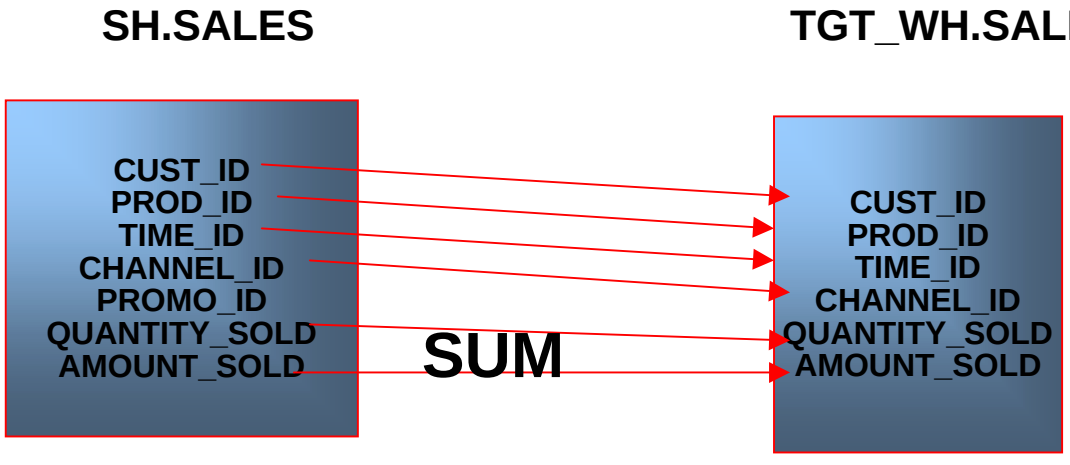
练习使用的数据源和目标说明

- **Oracle数据源**
  - SH用户下的CUSTOMERS和SALES表
- **文件数据源**
  - 存放于\exercise\CUST_JOB.txt
- **目标**
  - 用户是TGT_WH（用户创建命令见文件CREATE_TGT_WH.SQL）
  - TGT_CUST表（由ODI创建）
  - TGT_SALES表（预先创建好，DDL文件是CREATE_TGT_SALES.SQL）

数据源和目标的对应关系1_Interface1



数据源和目标的对应关系1_Interface2



ODI 开发设计

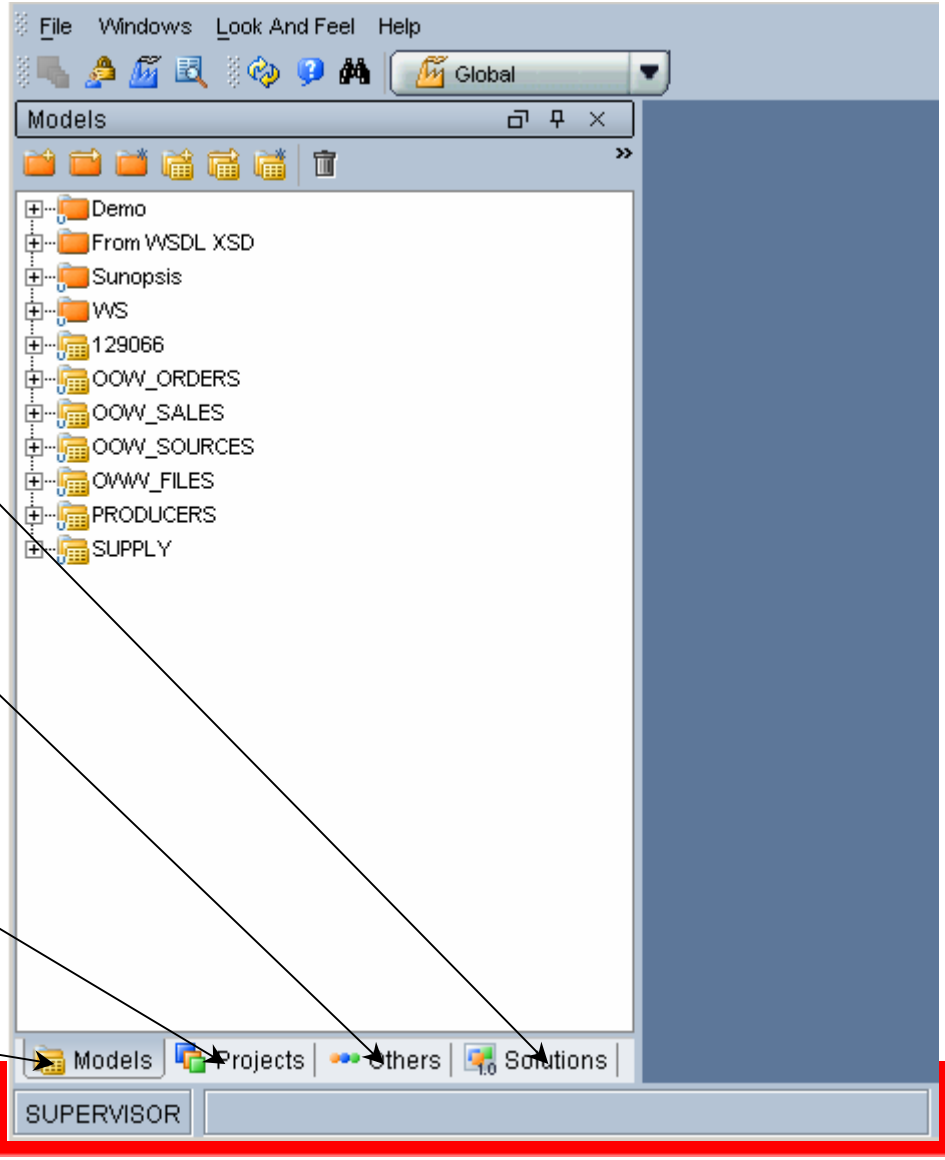
Designer 布局

高级版本控制

全局对象
(变量,
序列,
用户定义函数,
标记)

开发的内容

元数据



Models

什么是 Model?

Model – 描述关系型数据的模型. 是一组存放在特定的技术（如 Oracle）的 SCHEMA 的数据存储.

一个模型中包含的元数据可以通过反向工程（reverse-engineered） 从真实的环境导入, 也可以在 ODI 中创建.

什么是反向工程(Reverse-Engineering)?

Reverse-Engineering – 一种自动检索元数据并且在ODI中创建或同步模型的方法.

DBMS 的反响工程的方法

- 标准的 反向工程
  - 使用 **JDBC** 连接的特性检索元数据，然后写到 ODI 资料档案库.
  - 需要和实的驱动程序
- 自定义 反向工程
  - 从应用/数据库的数据字典中读取元数据写到 ODI 资料档案库
  - 使用技术专用的策略, 通过Reverse-engineering Knowledge Module (RKM)实现

Projects

Projects

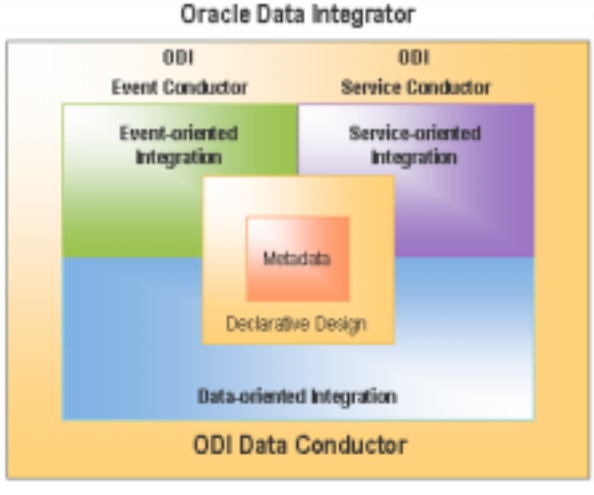
- 项目包含开发人员开发的所有的对象
- 项目的原子元素包括：
  - 接口 (定义将数据转换加载到表中的规则)
  - 过程 (预定义的步骤，用来执行 SQL, Jython 或其他脚本)
  - 变量
  - 用户定义函数

Projects: 执行单元

- 尽管是原子元素，如接口，程序等也可以单独执行，他们通常都被组织成工作流来执行
- 这个工作流就是包（Package）.
- 包/接口的代码可以被固化用来发布到生产环境, 这就是 方按（Senario）


EXERCISE

练习2： Model & KM


Oracle Data Integrator
ODI
Event Conductor
ODI
Service Conductor
Event-oriented Integration
Service-oriented Integration
Metadata
Declarative Design
Data-oriented Integration
ODI Data Conductor

Interface 的概念

什么是 Interface?

Interface – 一种 ODI 对象，用来从一个或多个源数据存储往一个目标数据存储加在数据. 一个 interface 实现声明式的规则：映射，过滤，关联和约束.

接口的声名式规则

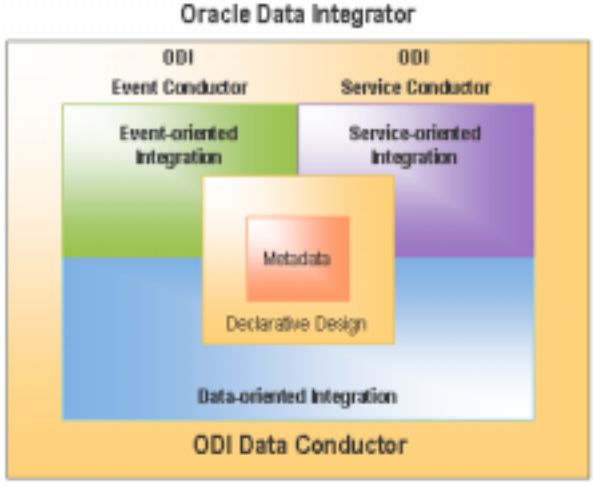
- 接口中的声名式规则通过如下方式实现:
  - 映射
  - 过滤
  - 关联
  - 约束
- 这些规则在 **Designer** 的下列对象中定义:
  - **models**
  - **Interfaces**
- 这些规则存储在 work repository

在那里定义声名式规则?

声名式规则在下面的这些地方定义:

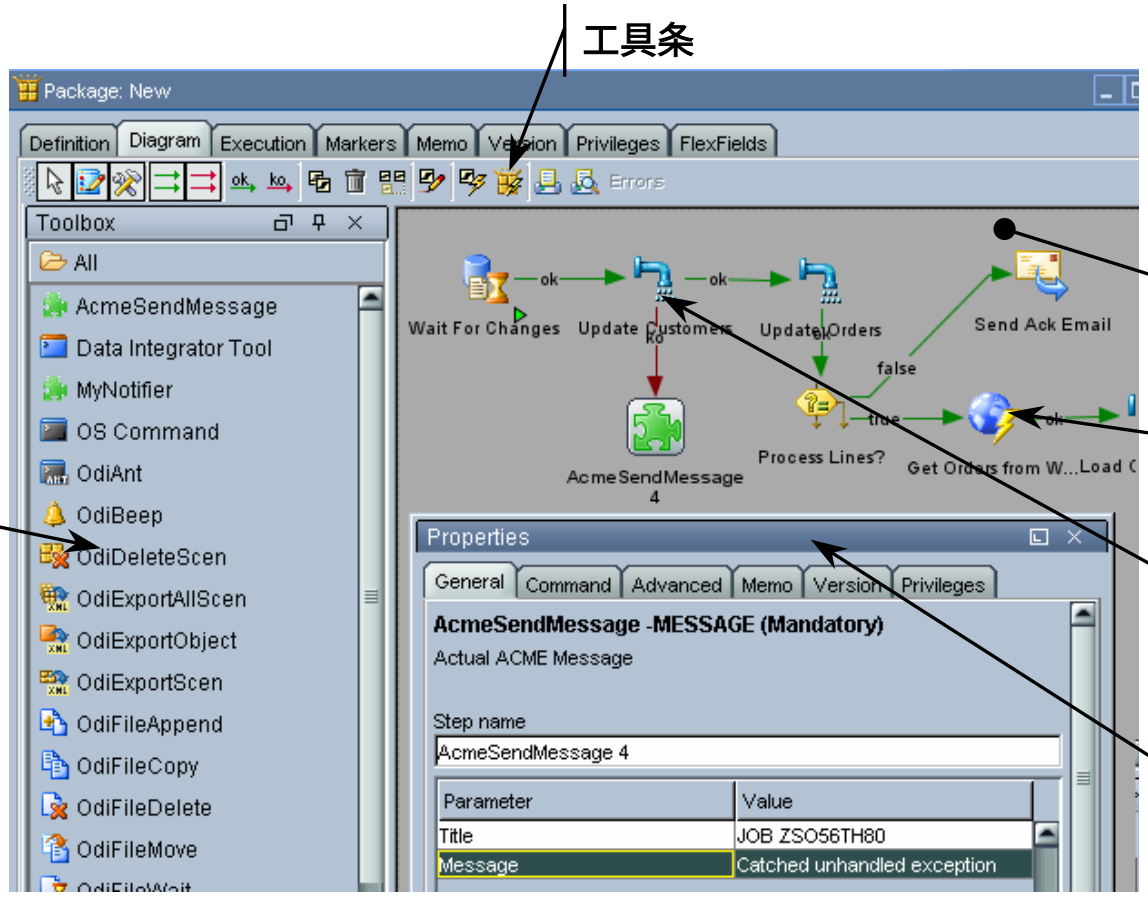
映射	interface diagram. 目标数据存储的每一列是一个映射.
过滤	interface diagram. 可以从 Model 中定义的过滤带进来也可以在 Interface 中手工定义.
关联	interface diagram.可以从 Model 中定义的过滤带进来也可以在 Interface 中手工定义.
约束	在目标数据存储的 model 中定义. 在 interface 的 tab 中可以选择要校验的约束.

练习3： Interface



Packages概览

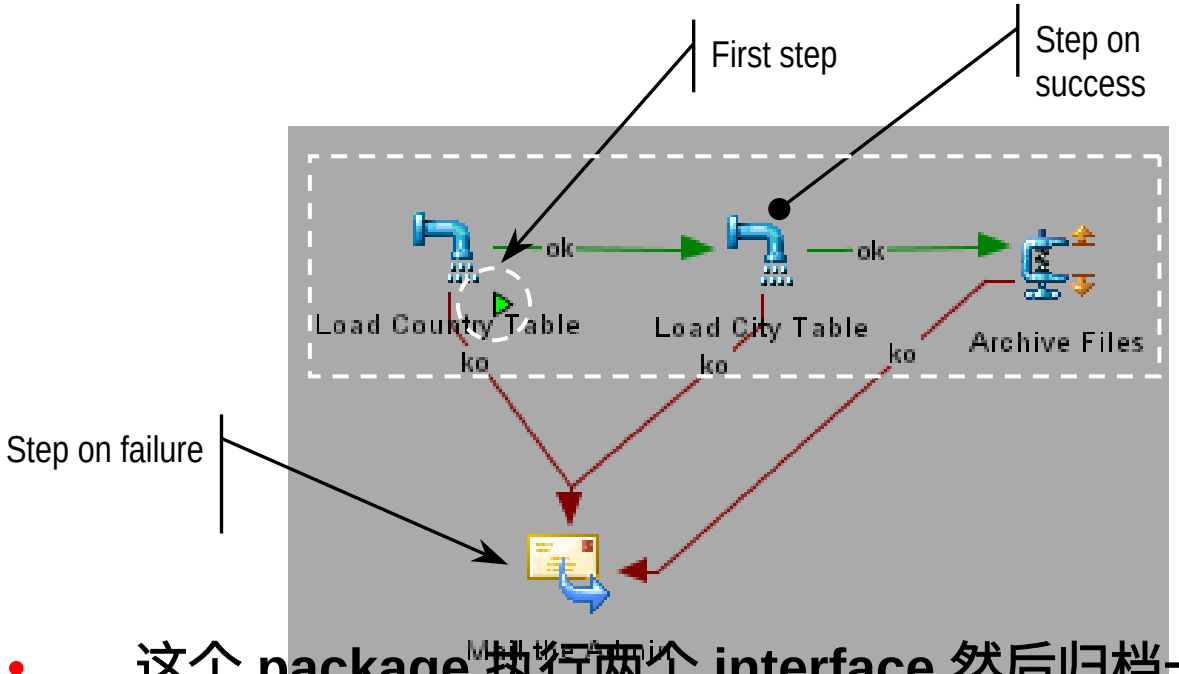
Package 的界面


工具条
Package: New
Definition
Diagram
Execution
Markers
Memo
Version
Privileges
FlexFields
Errors
Toolbox
All
AcmeSendMessage
Data Integrator Tool
MyNotifier
OS Command
OdiAnt
OdiBeep
OdiDeleteScen
OdiExportAllScen
OdiExportObject
OdiExportScen
OdiFileAppend
OdiFileCopy
OdiFileDelete
OdiFileMove
Wait For Changes
Update Customers
Update Orders
Send Ack Email
AcmeSendMessage 4
Process Lines?
Get Orders from W...
ok
ko
false
true
Properties
General
Command
Advanced
Memo
Version
Privileges
AcmeSendMessage -MESSAGE (Mandatory)
Actual ACME Message
Step name
AcmeSendMessage 4
Parameter
Value
Title
JOB ZSO56TH80
Message
Catched unhandled exception
Diagram
ODI tool step
Interface step
Properties of selected step
Toolbox for ODI tools

序列化的步骤

- 必须定义第一步
  - Right click > *First Step*
- 每个步骤执行后分成两个方向:
  - 成功: *ok* (返回代码 0)
  - 失败: *ko* (返回代码非 0)

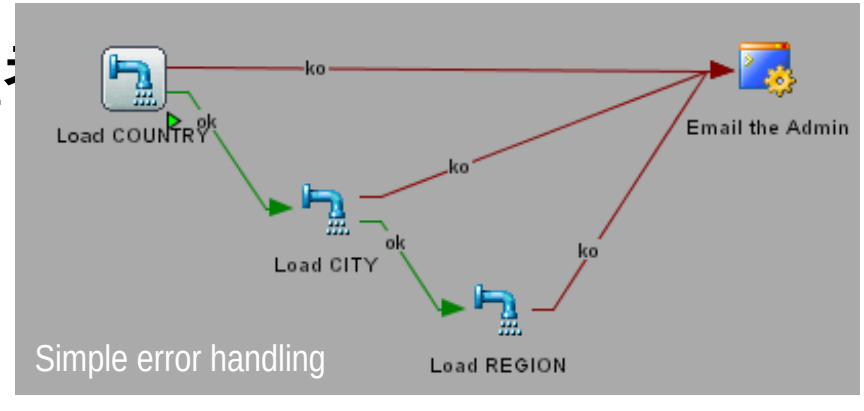
一个简单的 Package



- 这个 package 执行两个 interface 然后归档一些文件.
- 如果任何一个步骤失败则向管理员发送 email 并中止.

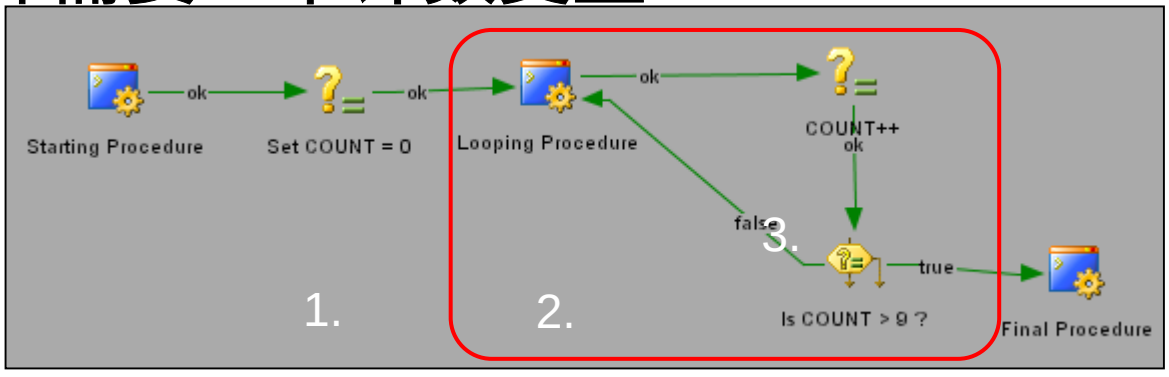
错误处理

- 接口失败：发生了致命的错误或者错误数超出限制.
- 过程和其他步骤失败：发生了致命的错误
- 尽量



如何创建循环逻辑

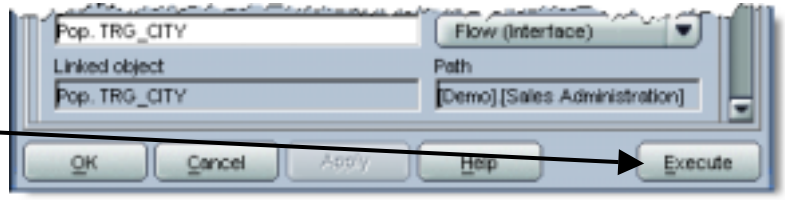
- 循环需要一个计数变量



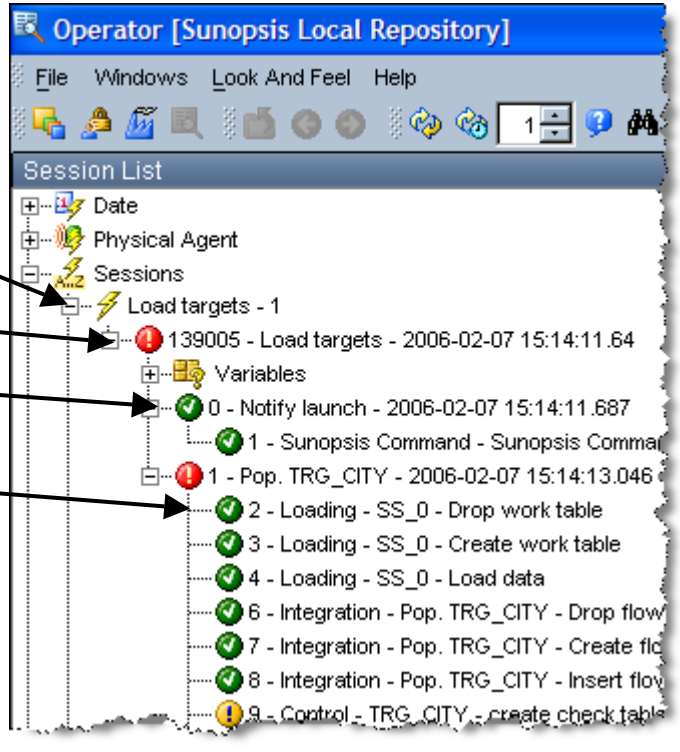
1. 给变量设置一个初始值
2. 执行一个或多个要重复执行的步骤
3. 执行完步骤之后增加计数器
4. 评估计数器决定是否回到第2步还是结束循环

执行 Package

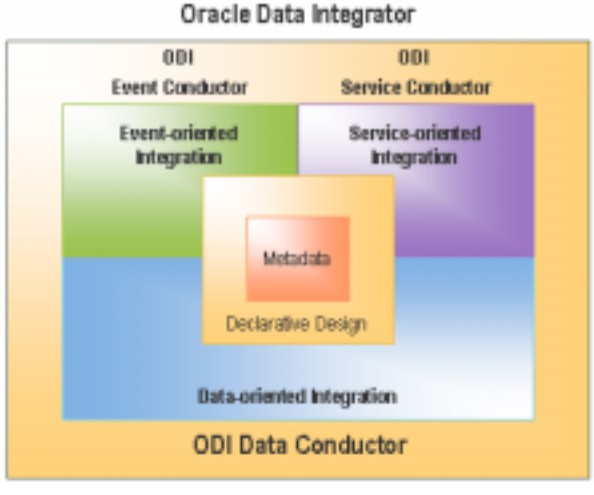
1. 在 **Package** 的窗口点击执行按钮



2. 打开 ***Operator***
   - package 会作为 **session** 执行
     - 每个 package 步骤是一个会话 **step**
       - 工具步骤在 operator 中显示成一个单独的 **task**
       - Interface 的子步骤将每个命令显示成为一个单独的 **task**



练习4&5： Package & Scenario



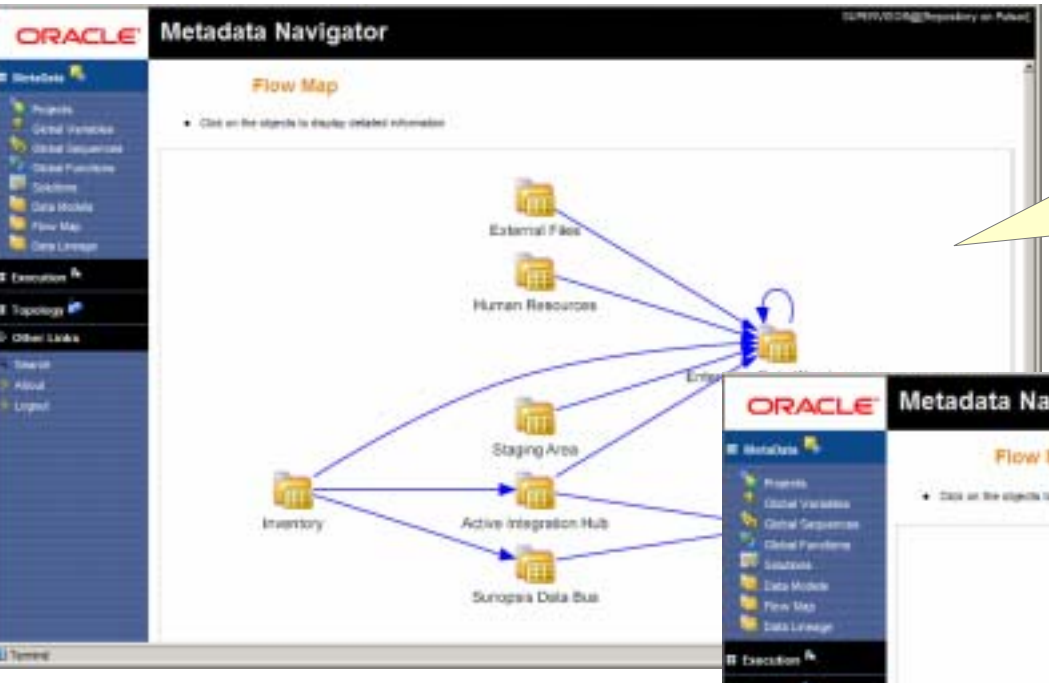
Metadata Navigator (Web 界面)

Metadata Navigator

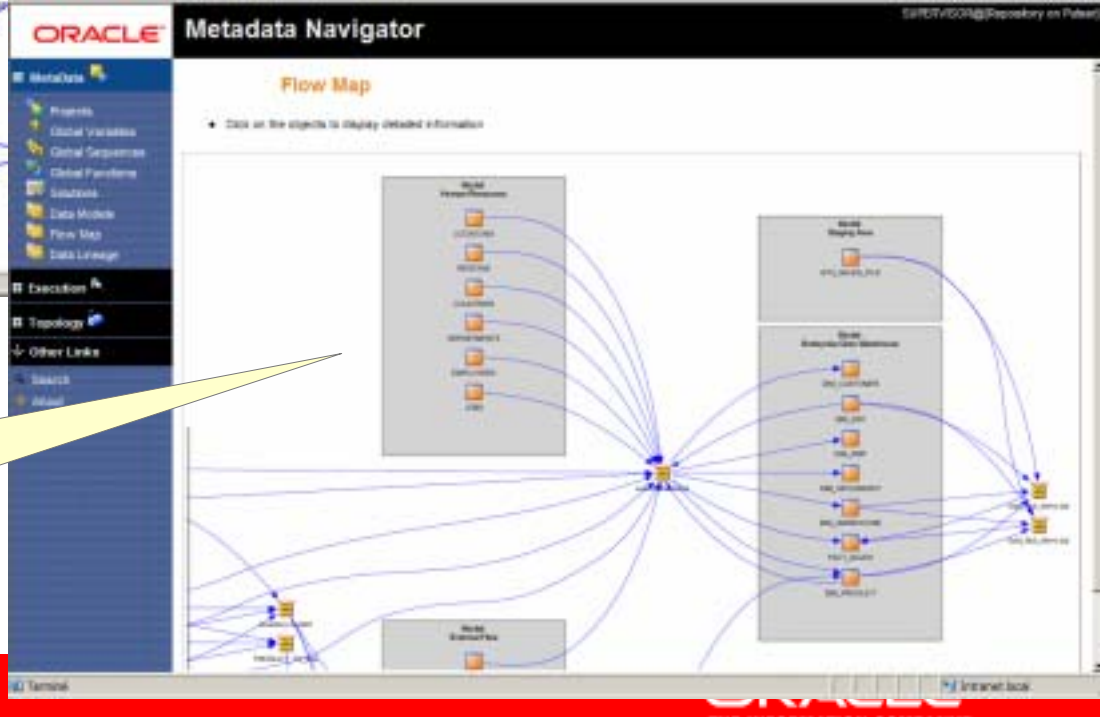
- 特性
  - 信息系统的数据流地图
  - 基于声明式规则的动态的元数据影响分析
  - 映射的关系图
- 好处
  - 企业元数据的统一视图
  - 企业数据集成流的统一视图
  - 钻取到从源到目标的转换
  - 基于业务理解数据集成流程

Metadata Navigator: Flow Map

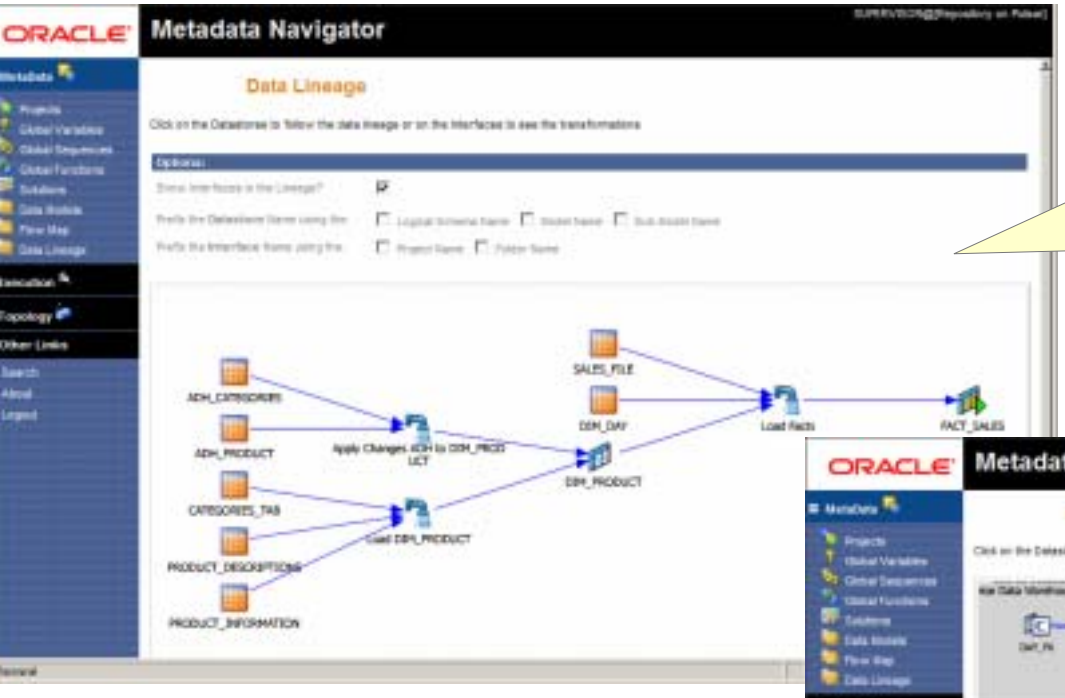
Metadata Navigator displays flows between the applications of the Information System based on the interfaces, transformations, packages and scenarios in production.



Metadata Navigator can also display more detailed maps of the flows at the datastore level

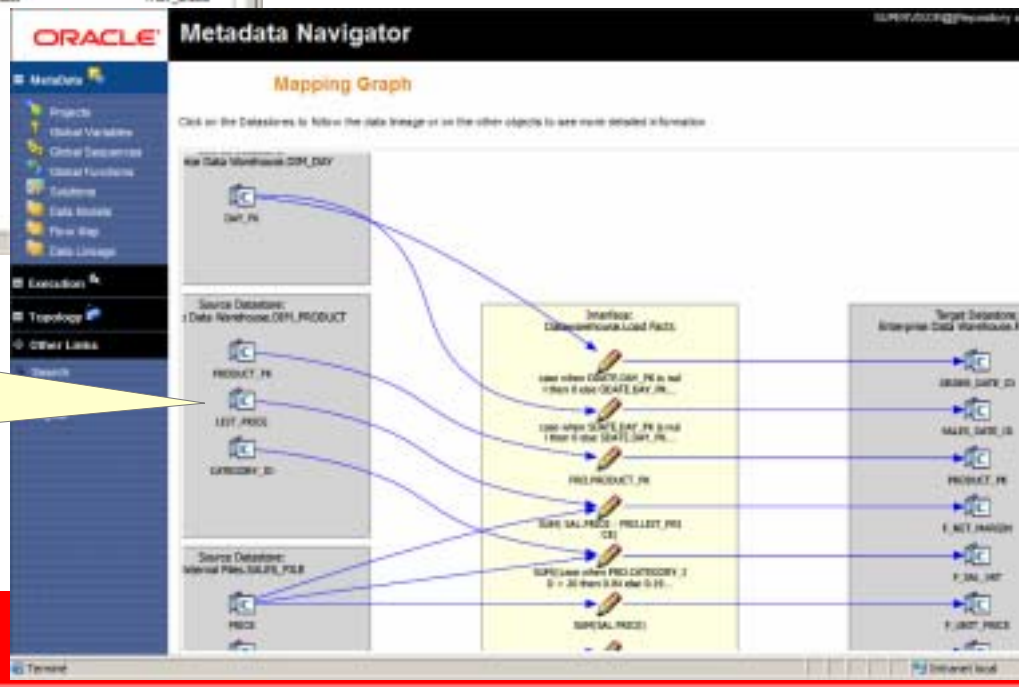


Metadata Navigator: Data Lineage



Metadata Navigator allows drill-down on transformations to perform dynamic metadata lineage

The example shows a metadata lineage analysis of the FACT_SALES table.



Understanding the Descriptive Rules and Transformations is achieved by simply opening the interfaces and looking at the field mappings.


Q&A


ORACLE®
甲骨文