Uplatz

**training.uplatz.com**

Training Provider for IT and Certification Courses

# Bundle Course – ETL Tools – Course Syllabus

## Talend - Course Curriculum

### 1. Role of Open Source ETL Technologies in Big Data

- Overview on: TOS (Talend Open Studio) for Data Integration
- ETL concepts
- Data warehousing concepts

### 2. Talend

- Why Talend?
- Features
- Advantages
- Talend Installation/System Requirements
- GUI layout (designer)

- Understanding it's Basic Features
- Comparison with other market leader tools in ETL domain
- Important areas in Talend Architecture: Project
- Workspace
- Job
- Metadata
- Propagation
- Linking components

**3. Talend: Read & Write various Types of Source/Target System**

- Data Source Connection
- File as Source
- Create meta data
- Database as source
- Create metadata
- Using MySQL database (create tables, Insert, Update Data from Talend)
- Read and write into excel files, into multiple tabs
- View data
- How to capture log and navigate around basic errors
- Role of tLogrow and how it makes developers life easy

**4. Talend: How to Transform Your Business: Basic**

- Using Advanced components like: tMap, tJoin, tFilter, tSortRow, tAggregateRow, tReplicate, tSplit, Lookup, tRowGenerator

**5. Talend: How to Transform Your Business: Advanced 1**

- Trigger (types) and Row Types
- Context Variables (parameterization)

- Functions (basic to advanced functions to transform business rules such as string, date, mathematical etc.)
- Accessing job level / component level information within the job

**6. Talend: How to Transform Your Business: Advanced 2**

- Type Casting (convert data types among source-target platforms)
- Looping components (like tLoop, tFor)
- tFileList
- tRunJob
- How to schedule and run talend DI jobs externally (not in GUI)

**7. Working with Hierarchical File Structures**

- Read and Write an XML file, configure the schema and XPath expression to parse an XML file
- Read and Write a JSON file, configure the schema and JSONPath expression to parse a JSON file
- Read and write delimited, fixed width files.

**8. Context Variables and Global Variables**

- Create context/global variables
- Use context/global variables in the configuration of Talend components
- Load context variables from a flow

**9. Best practices**

- Working with databases and implementing data warehousing concepts
- Working with files (excel, delimited, JSON, XML etc.)

**10. Orchestration and Controlling Execution Flow**

- Files - Use components to list, archive, and delete files from a directory

- Database – Controlling Commit and Rollback
  - COMMIT at end of job/ every x number of rows
  - Rollback on error

**11. Shared DB connection across jobs and subjobs**

- Use triggers to connect components and subJobs
- Orchestrate several jobs in master jobs.
- Handling Errors
  - Kill a Job on a component error
  - Implement a specific Job execution path on a component error
  - Configure the log level in the console

statement of accounts

## SAP Data Services (BODS) - Course Syllabus

1) BODS overview

1. Over view of the Data services
2. Data services benefits, associated products, interfaces
3. Data services Architecture on single and distributed environment

2) BODS Designer concepts

1. Creating the repository (Local, Central repository
2. Exploring the menu options in the designer
3. Creating the project, Job flow, Data flow, Work flows etc., defining different types of Data stores (Source and destination data stores)
4. Use data store and system configurations
5. Defining file formats for flat, Excel, XML files

3) Batch Jobs

1. Creating Batch Jobs
2. Work with objects
3. Create a data flow

4. Adding the Query transform to the data flow
5. Use target tables
6. Execute the job

4) Defining Data Integrator Transforms

1. Date Generation Transform
2. Pivot Transform
3. Reverse Pivot Transform
4. XML_Pipeline Transform

5) Defining Data Platform Transforms

1. Query Transform
2. Case Transform
3. Merge Transform
4. Validation Transform
5. Row Generation Transform
6. SQL Transform

6) Defining Data Quality Transform

1. Address Cleanse
2. Geocoder

7) Implementation of SCD

1. SCD Type0
    - Query Transform
2. SCD Type1
    - Table Comparison Transform
    - Map_Operation Transform
3. SCD Type2
    - Table Comparison Transform
    - History Preserving Transform
    - Key Generation Transform

8) Using Functions, Scripts, and Variables

1. Define built-in functions
2. Use functions in expressions
3. Use variables and parameters
4. Create Custom functions

5. Use Data Services scripting language

9) Data Assessment

1. Using the data profiler
2. Using the validation transform

10) Setting up Error Handling

1. Set up recoverable work flows

11) Setting up Exception Handling

1. Try/Catch Techniques

12) IF Conditional

13) While-Loop Implementation

14) Capturing Changes in Data

15) Data Assessment

1. Update data over time
2. Use source-based CDC
3. Use target-based CDC
   - SCD Type 2

16) Multi-User Environment(Local Repo Vs Central Repo)

17) SAP Integration

1. Data extraction from ECC system to File, Table and SAP BI
2. Data extraction from ECC Extractors
3. Data Extraction from SAP BW
4. ABAP Workflows

18) SAP HANA and SAP BODS Integration

1. Introduction to SAP HANA
2. Introduction to SAP HANA Studio
3. Create Data Store for SAP HANA
4. Perform the load into SAP HANA
5. Store the data in Column Store
6. Preview the data in SQL console of SAP HANA Studio

19) Information Steward

1. Data Insight
2. Metadata management
3. Metapedia

**Oracle PL/SQL Course Syllabus**

**Description:** Introduction

**ORACLE PL-SQL PROGRAMMING - COURSE CONTENT**

- Course Objectives
- Course Agenda
- Describe the ER Schema
- PL/SQL development environments available in this course
- Introduction to SQL Developer

Working with Oracle Cloud Exadata Express Cloud Service

- Introduction to Oracle Database Exadata Express Cloud Service
- Accessing Cloud Database using SQL Workshop
- Connecting to Exadata Express using Database Clients

Introduction to PL/SQL

- Overview of PL/SQL
- Identify the benefits of PL/SQL Subprograms
- Overview of the types of PL/SQL blocks
- Create a Simple Anonymous Block
- How to generate output from a PL/SQL Block?

Declare PL/SQL Variables

- List the different Types of Identifiers in a PL/SQL subprogram
- Usage of the Declarative Section to Define Identifiers
- Use variables to store data
- Identify Scalar Data Types
- The %TYPE Attribute
- What are Bind Variables?
- Sequences in PL/SQL Expressions

Write Anonymous PL/SQL Blocks

- Describe Basic PL/SQL Block Syntax Guidelines
- Learn to Comment the Code
- Deployment of SQL Functions in PL/SQL
- How to convert Data Types?
- Describe Nested Blocks
- Identify the Operators in PL/SQL

SQL Statements in a PL/SQL block

- Invoke SELECT Statements in PL/SQL
- Retrieve Data in PL/SQL
- SQL Cursor concept
- Avoid Errors by using Naming Conventions when using Retrieval and DML Statements
- Data Manipulation in the Server using PL/SQL
- Understand the SQL Cursor concept
- Use SQL Cursor Attributes to Obtain Feedback on DML
- Save and Discard Transactions

Control Structures

- Conditional processing using IF Statements
- Conditional processing using CASE Statements
- Describe simple Loop Statement
- Describe While Loop Statement
- Describe For Loop Statement
- Use the Continue Statement

Composite Data Types

- Use PL/SQL Records
- The %ROWTYPE Attribute
- Insert and Update with PL/SQL Records
- INDEX BY Tables
- Examine INDEX BY Table Methods
- Use INDEX BY Table of Records

Explicit Cursors

- What are Explicit Cursors?
- Declare the Cursor
- Open the Cursor
- Fetch data from the Cursor
- Close the Cursor
- Cursor FOR loop
- The %NOTFOUND and %ROWCOUNT Attributes
- Describe the FOR UPDATE Clause and WHERE CURRENT Clause

Exception Handling

- Understand Exceptions
- Handle Exceptions with PL/SQL
- Trap Predefined Oracle Server Errors
- Trap Non-Predefined Oracle Server Errors
- Trap User-Defined Exceptions
- Propagate Exceptions
- RAISE_APPLICATION_ERROR Procedure

Stored Procedures

- Create a Modularized and Layered Subprogram Design

- Modularize Development With PL/SQL Blocks
- Understand the PL/SQL Execution Environment
- List the benefits of using PL/SQL Subprograms
- List the differences between Anonymous Blocks and Subprograms
- Create, Call, and Remove Stored Procedures
- Implement Procedures Parameters and Parameters Modes
- View Procedure Information

Stored Functions

- Create, Call, and Remove a Stored Function
- Identify the advantages of using Stored Functions
- Identify the steps to create a stored function
- Invoke User-Defined Functions in SQL Statements
- Restrictions when calling Functions
- Control side effects when calling Functions
- View Functions Information

Debugging Subprograms

- How to debug Functions and Procedures?
- Debugging through SQL Developer

Packages

- Listing the advantages of Packages
- Describe Packages
- What are the components of a Package?
- Develop a Package
- How to enable visibility of a Packages Components?
- Create the Package Specification and Body using the SQL CREATE Statement and SQL Developer
- Invoke the Package Constructs
- View the PL/SQL Source Code using the Data Dictionary

Deploying Packages

- Overloading Subprograms in PL/SQL
- Use the STANDARD Package
- Use Forward Declarations to solve Illegal Procedure Reference

- Implement Package Functions in SQL and Restrictions
- Persistent State of Packages
- Persistent State of a Package Cursor
- Control side effects of PL/SQL Subprograms
- Invoke PL/SQL Tables of Records in Packages

Implement Oracle-Supplied Packages in Application Development

- What are Oracle-Supplied Packages?
- Examples of some of the Oracle-Supplied Packages
- How does the DBMS_OUTPUT Package work?
- Use the UTL_FILE Package to Interact with Operating System Files
- Invoke the UTL_MAIL Package
- Write UTL_MAIL Subprograms

Dynamic SQL

- The Execution Flow of SQL
- What is Dynamic SQL?
- Declare Cursor Variables
- Dynamically Executing a PL/SQL Block
- Configure Native Dynamic SQL to Compile PL/SQL Code
- How to invoke DBMS_SQL Package?
- Implement DBMS_SQL with a Parameterized DML Statement
- Dynamic SQL Functional Completeness

Design Considerations for PL/SQL Code

- Standardize Constants and Exceptions
- Understand Local Subprograms
- Write Autonomous Transactions
- Implement the NOCOPY Compiler Hint
- Invoke the PARALLEL_ENABLE Hint
- The Cross-Session PL/SQL Function Result Cache
- The DETERMINISTIC Clause with Functions
- Usage of Bulk Binding to Improve Performance

Triggers

- Describe Triggers