ODI JDBC ACCESS

BY ODI EXPERTS

V1.0 DOCUMENTATION

CONTENTS

READING AND WRITING ACCESS WITHOUT ODBC

The ODI JDBC ACCESS driver is created to Read/Write Access DB. This driver supports Access 2000 and later only. This driver works in all OS and does not need any DSN or ODBC and is completely written in Java.

LIMITATION OF THIS DRIVER

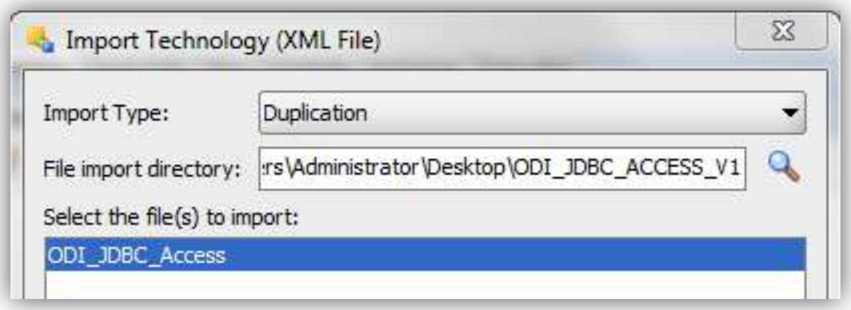
- In ODI Data / View Data on the data store will not work
- The datatype GUID/Replication ID is not supported
- Foreign Keys , Indexes and CKM is presently not supported.(This might be a feature in future release)

PRELIMINARY STEPS

- Extract the ODI JDBC ACCESS zip file and you will find these files
    - KM_IKM_ACCESS_INSERT/UPDATE_V1.xml
    - KM_LKM_ACCESS_to_SQL_V1.xml
    - KM_RKM_ACCESS_V1.xml
    - ODI_JDBC_ACCESS_V1.jar
    - TECH_ODI_JDBC_ACCESS.xml

- Copy the ODI_JDBC_ACCESS_v1.jar under your ODI Agent's Driver Folder and restart your Agent .For local, please copy in to your respective USERLIB Folder.

- Next Import the ODI JDBC ACCESS Technology in Duplication Mode.



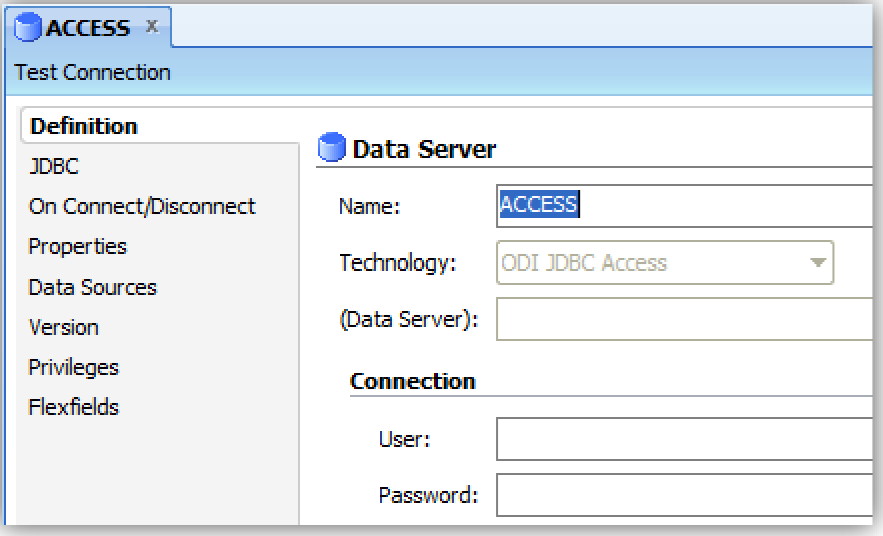
Import Type: Duplication

File import directory: :rs\Administrator\Desktop\ODI_JDBC_ACCESS_V1

Select the file(s) to import:

ODI_JDBC_Access

- Create a DataServer with just an appropriate Name for it. No Need to provide any other details.



Name: ACCESS

Technology: ODI JDBC Access

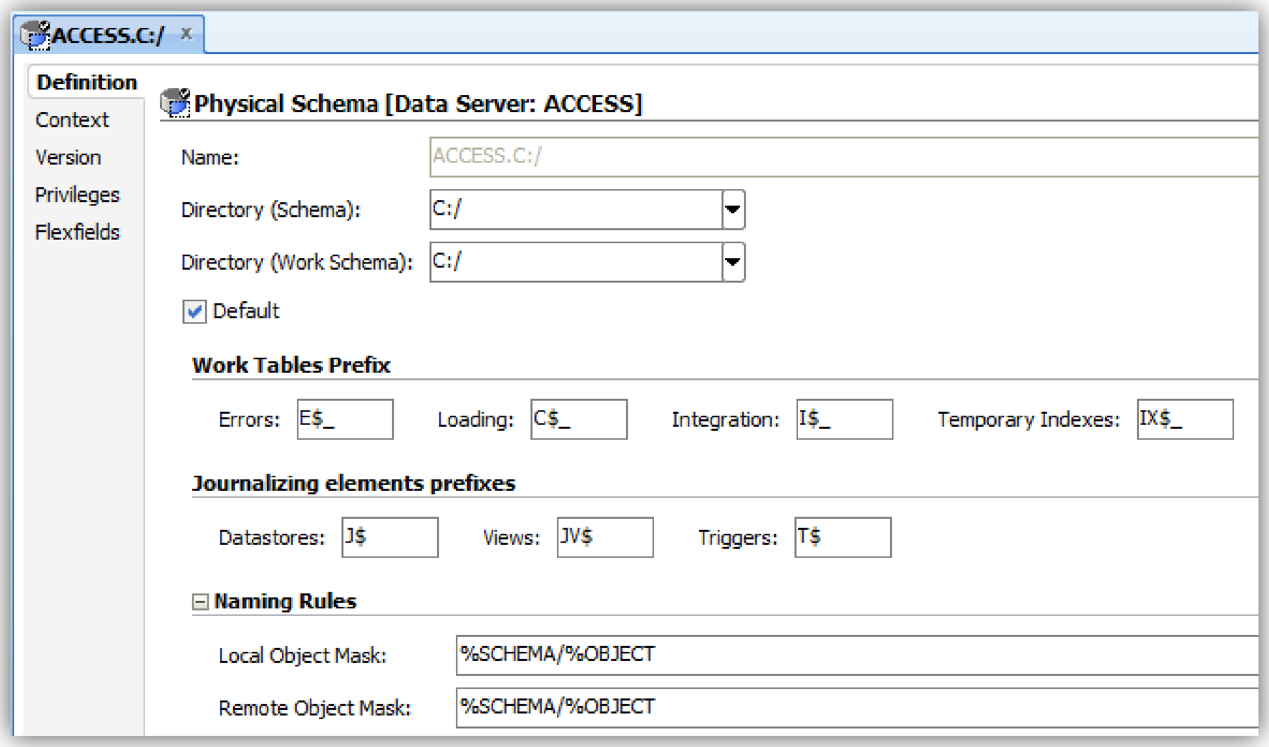
(Data Server):

User:

Password:

- Create a Physical Schema and provide the Directory where the ACCESS DBs are present.

[Note: - You can create ‘n’ Number of Physical and Logical Schema, under same Data Server]


ACCESS.C:/

Definition
Context
Version
Privileges
Flexfields

Physical Schema [Data Server: ACCESS]

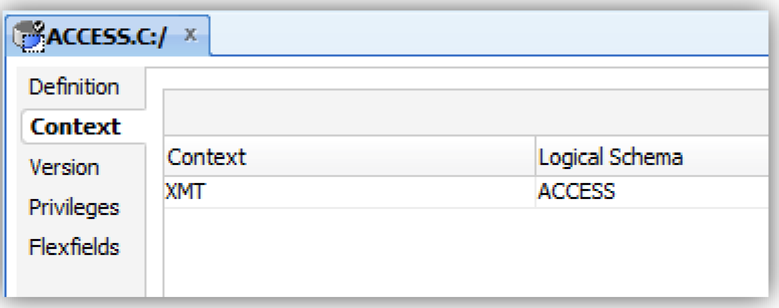
Name: ACCESS.C:/
Directory (Schema): C:/
Directory (Work Schema): C:/
☑ Default

Work Tables Prefix
Errors: E$_ Loading: C$_ Integration: I$_ Temporary Indexes: IX$_

Journalizing elements prefixes
Datastores: J$ Views: JV$ Triggers: T$

Naming Rules
Local Object Mask: %SCHEMA/%OBJECT
Remote Object Mask: %SCHEMA/%OBJECT

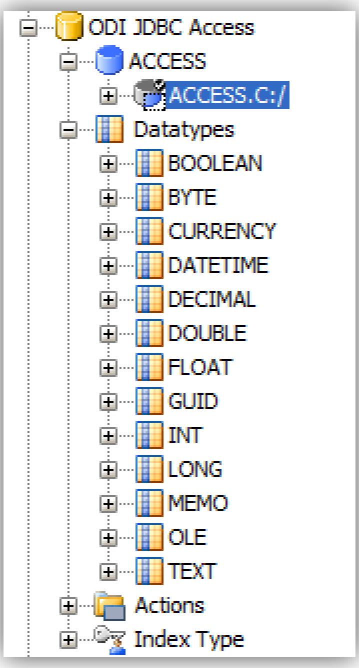
- Finally create a Logical Schema and link them through the appropriate Context.


ACCESS.C:/

Definition
Context
Version
Privileges
Flexfields

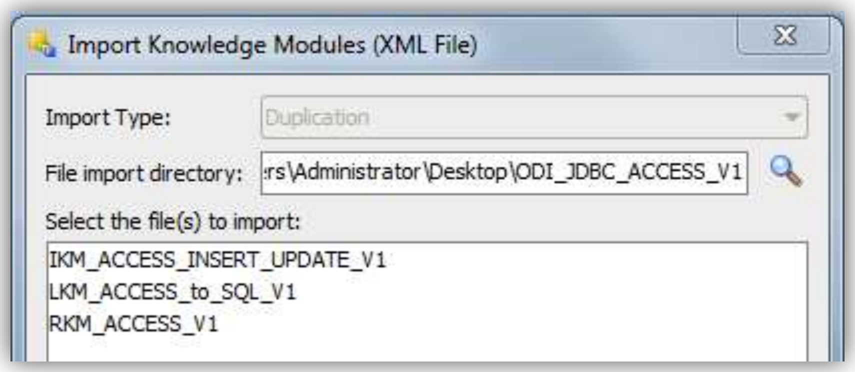
Context	Logical Schema
XMT	ACCESS

- Now you are ready to use this Technology.



ODI JDBC Access
ACCESS
ACCESS.C:/
Datatypes
BOOLEAN
BYTE
CURRENCY
DATETIME
DECIMAL
DOUBLE
FLOAT
GUID
INT
LONG
MEMO
OLE
TEXT
Actions
Index Type

- Login into Designer and Import the RKM, LKM and IKM respectively.



Import Knowledge Modules (XML File)

Import Type: Duplication

File import directory: :rs\Administrator\Desktop\ODI_JDBC_ACCESS_V1

Select the file(s) to import:

IKM_ACCESS_INSERT_UPDATE_V1
LKM_ACCESS_to_SQL_V1
RKM_ACCESS_V1

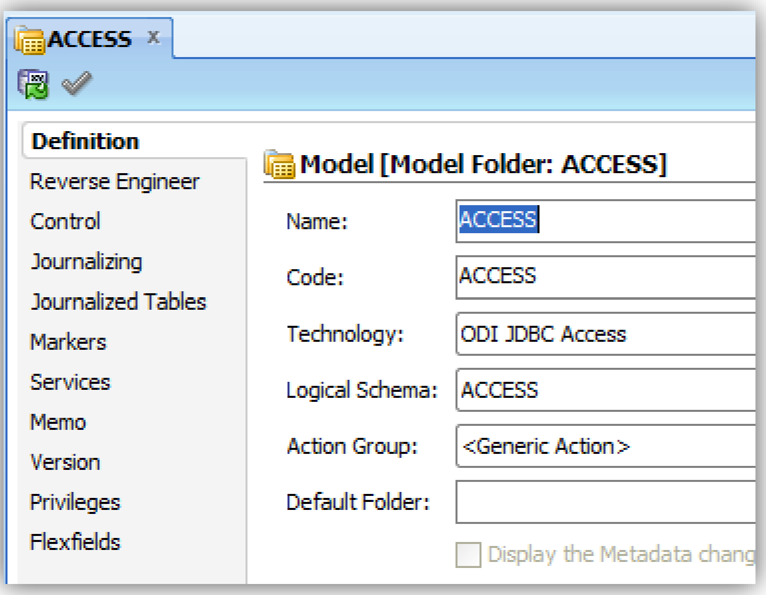
REVERSING THE ACCESS DB (RKM)

The different options of the RKM are

TABLE_NAME –If one of the ACCESS table is to be reversed then specify the Table Name. To reverse all the tables please use **%**.

[Note: - FK and Other Indexes are not supported presently]

Model Setup



Model [Model Folder: ACCESS]

Name:	ACCESS
Code:	ACCESS
Technology:	ODI JDBC Access
Logical Schema:	ACCESS
Action Group:	<Generic Action>
Default Folder:	

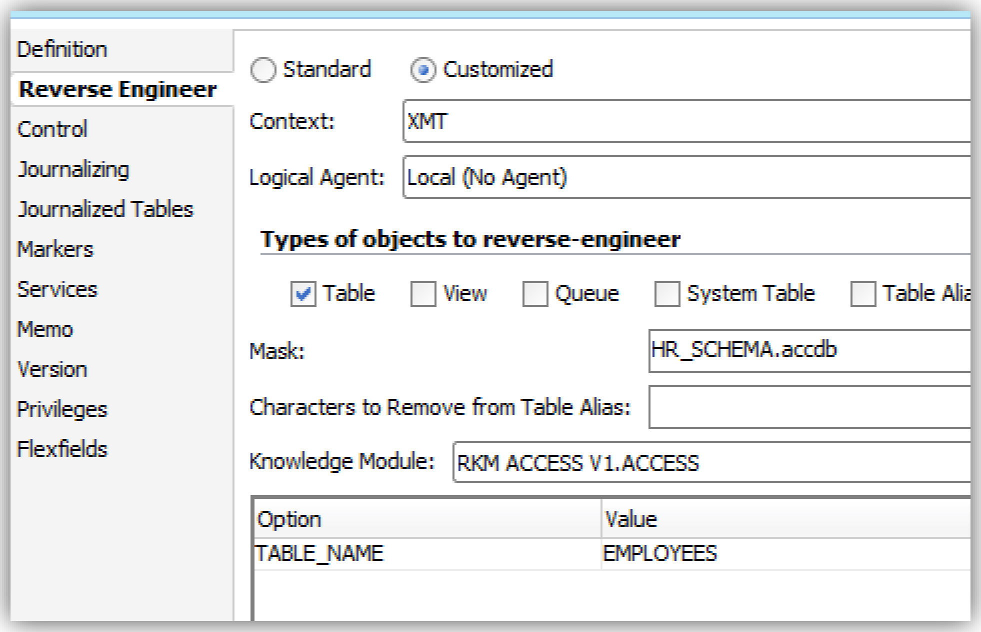
☐ Display the Metadata chang

REVERSE SINGLE TABLE

REVERSE ENGINEER – CUSTOMIZED

KNOWLEDGE MODULE – RKM ACCESS_V1

- MASK - << File Name with Extension>>
- TABLE_NAME – table name of the access.



Standard [✓ Customized]

Context: XMT

Logical Agent: Local (No Agent)

Types of objects to reverse-engineer

[✓ Table] View Queue System Table Table Alia

Mask: HR_SCHEMA.accdb

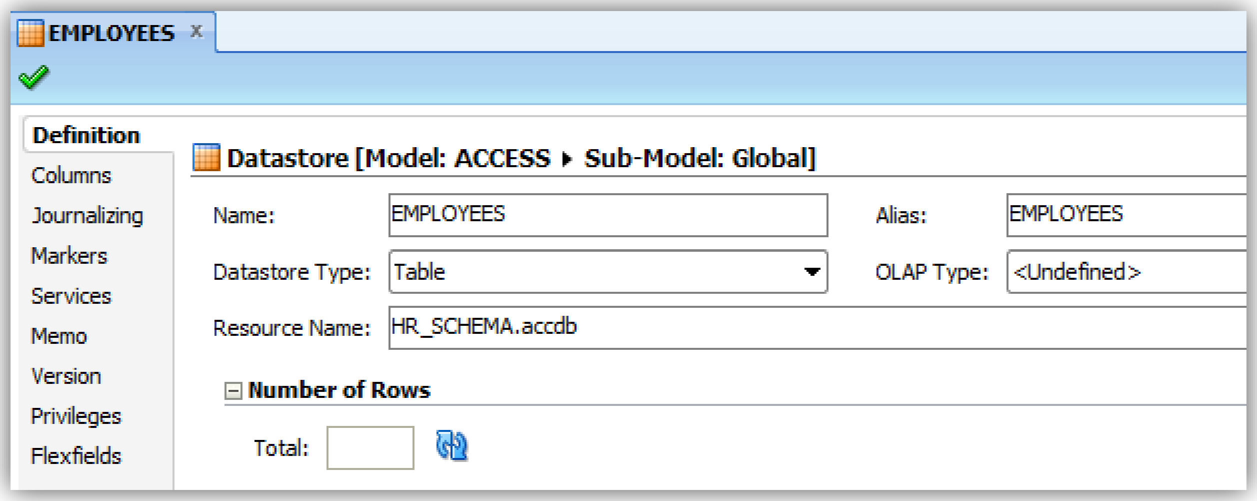
Characters to Remove from Table Alias:

Knowledge Module: RKM ACCESS V1.ACCESS

Option	Value
TABLE_NAME	EMPLOYEES

Once reversed the datastore will be similar as shown below,

- Name - << Access Table Name >>
- Resource Name – ACCESS DB Name with Extension



Datastore [Model: ACCESS ▸ Sub-Model: Global]

Name: EMPLOYEES

Alias: EMPLOYEES

Datastore Type: Table

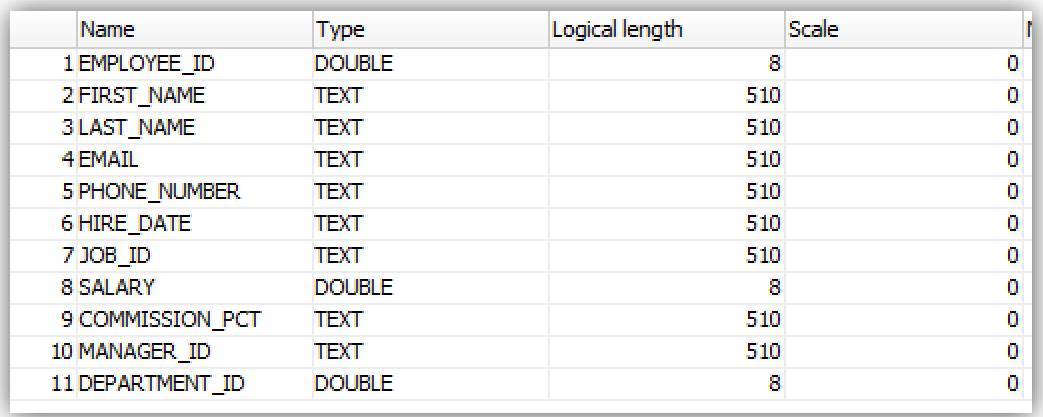
OLAP Type: <Undefined>

Resource Name: HR_SCHEMA.accdb

Number of Rows

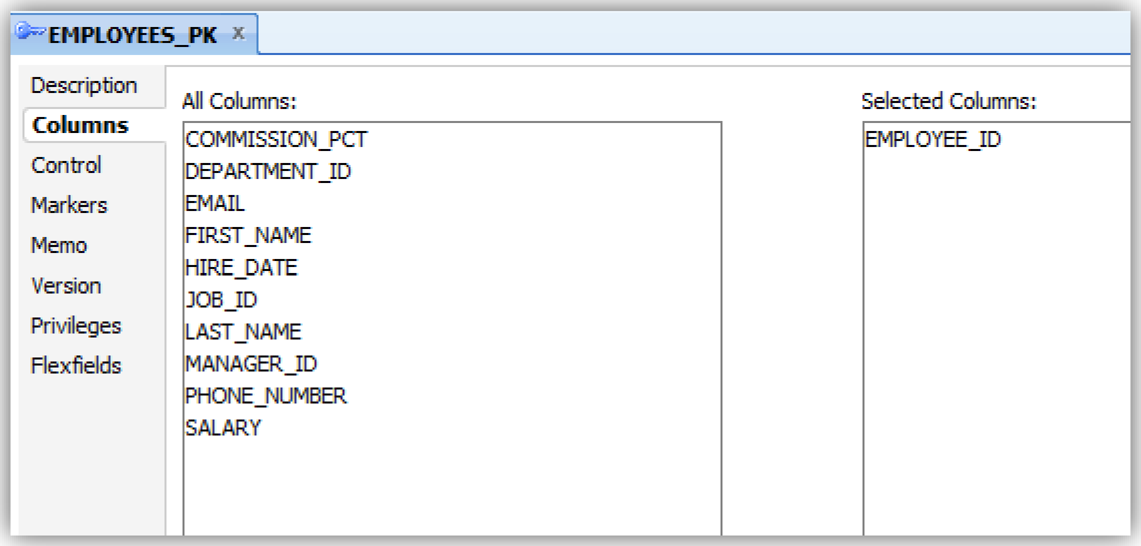
Total:

Data type is detected depending on the content.



	Name	Type	Logical length	Scale
1	EMPLOYEE_ID	DOUBLE	8	0
2	FIRST_NAME	TEXT	510	0
3	LAST_NAME	TEXT	510	0
4	EMAIL	TEXT	510	0
5	PHONE_NUMBER	TEXT	510	0
6	HIRE_DATE	TEXT	510	0
7	JOB_ID	TEXT	510	0
8	SALARY	DOUBLE	8	0
9	COMMISSION_PCT	TEXT	510	0
10	MANAGER_ID	TEXT	510	0
11	DEPARTMENT_ID	DOUBLE	8	0

Primary Key is also detected



All Columns:

COMMISSION_PCT
DEPARTMENT_ID
EMAIL
FIRST_NAME
HIRE_DATE
JOB_ID
LAST_NAME
MANAGER_ID
PHONE_NUMBER
SALARY

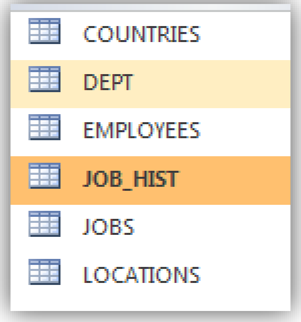
Selected Columns:

EMPLOYEE_ID

[Note: - FK and Other Indexes are not supported presently]

REVERSE ALL THE TABLES IN ACCESS

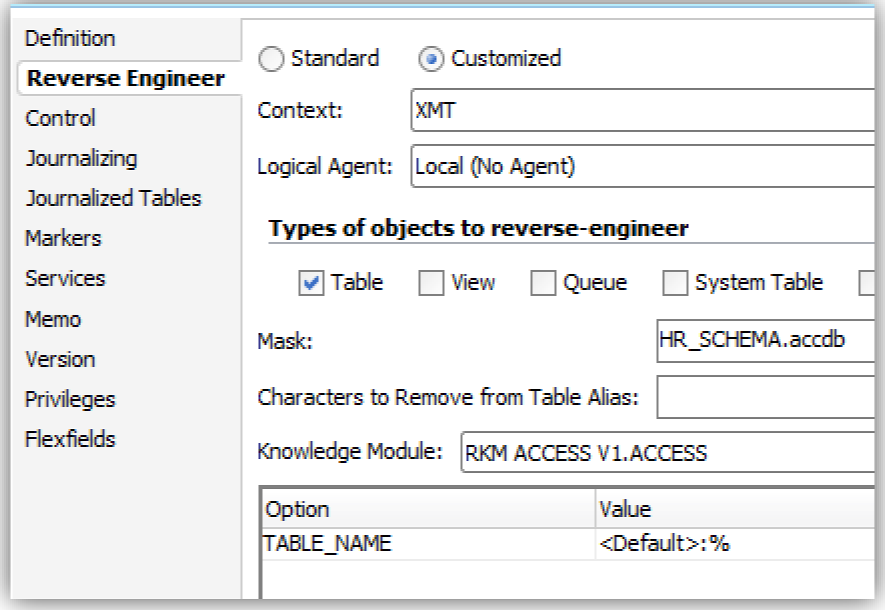
SAMPLE ACCESS DB



COUNTRIES
DEPT
EMPLOYEES
JOB_HIST
JOBS
LOCATIONS

REVERSE ENGINEER – CUSTOMIZED

KNOWLEDGE MODULE – RKM ACCESS_V1



Standard [✓ Customized]

Context: XMT

Logical Agent: Local (No Agent)

Types of objects to reverse-engineer

[✓ Table] View Queue System Table

Mask: HR_SCHEMA.accdb

Characters to Remove from Table Alias:

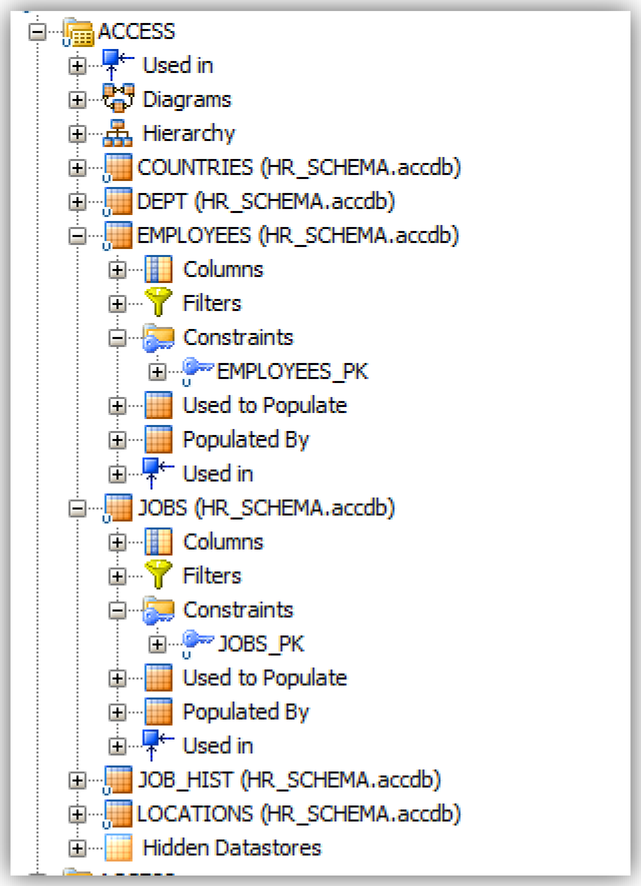
Knowledge Module: RKM ACCESS V1.ACCESS

Option	Value
TABLE_NAME	<Default>:%

- MASK - << File Name with Extension>>
- TABLE_NAME – %.

[NOTE: - T0 reverse all the tables in the access, please use %]

As you can see from the below image, all the tables are reversed including PK.



LOADING FROM ACCESS TO DB (LKM)

The different options of LKM are

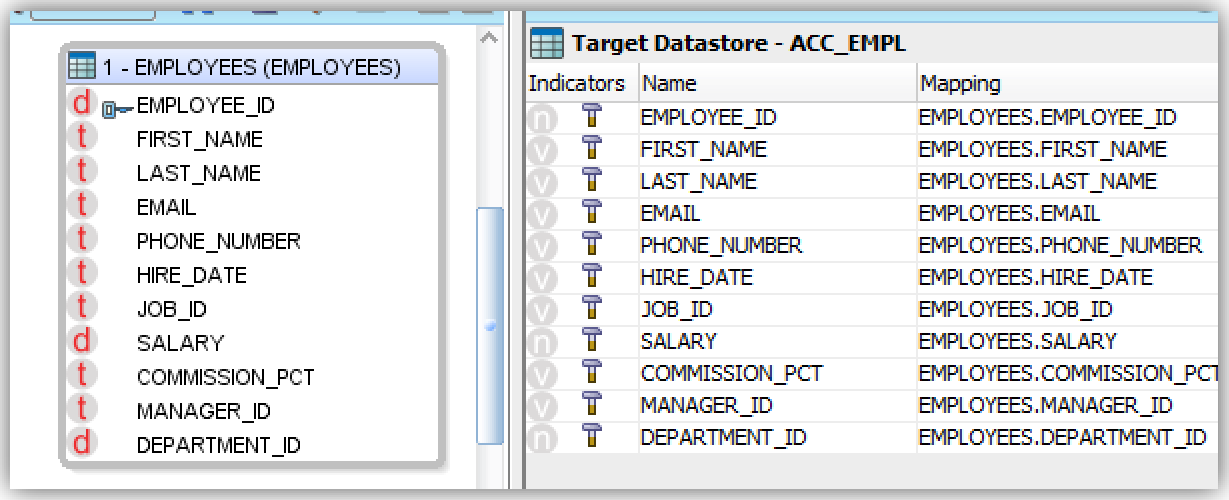
DELETE_TEMPORARY_OBJECTS: As present in all the KM options, this option is to retain C$.

Few Restrictions

- Filters has to be only on the Staging
- Joins between two Access Tables also have to be on Staging
- Any SQL Operation on source has to be performed on the Staging or Target side only.

LOADING ACCESS DB INTO DB

Map the column accordingly.



1 - EMPLOYEES (EMPLOYEES)

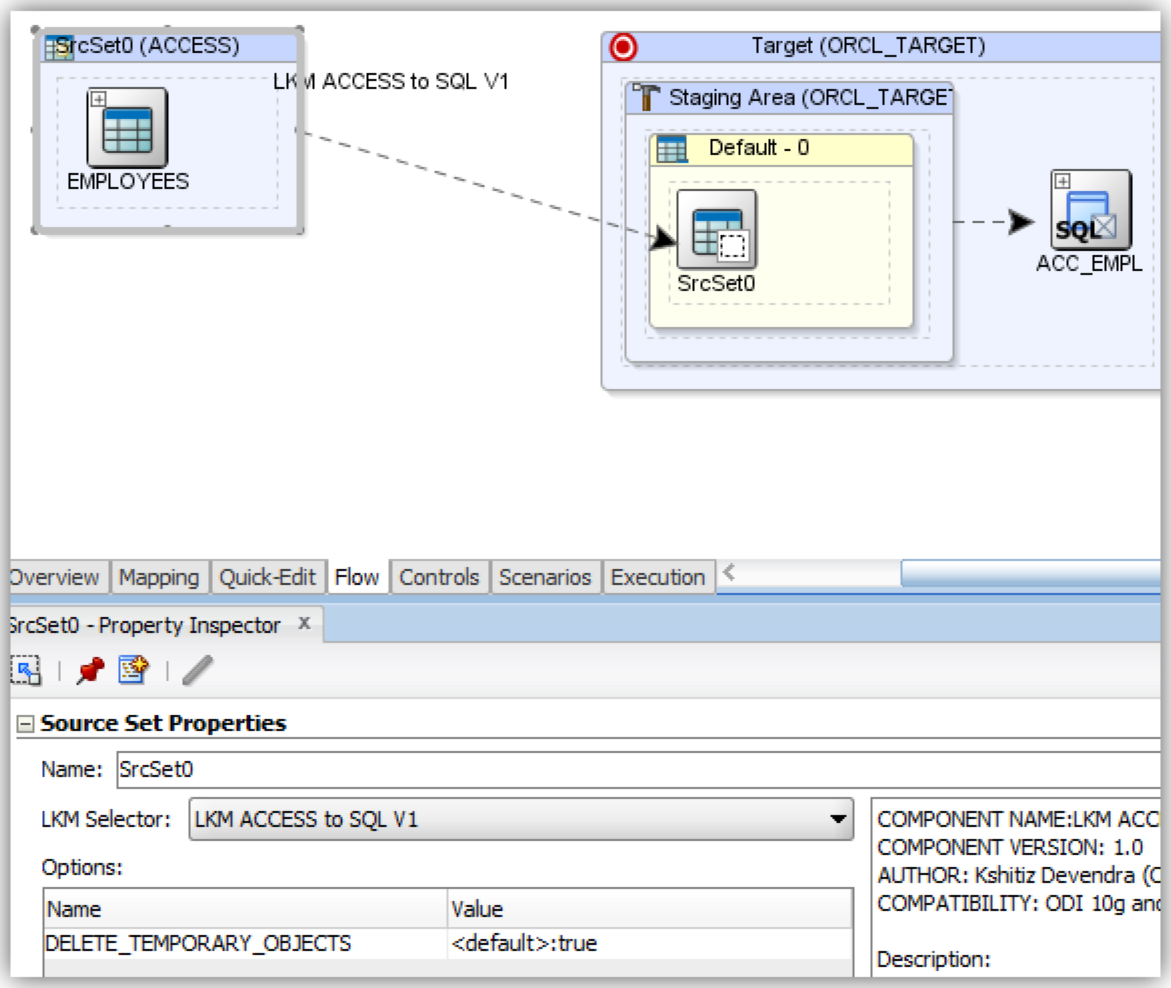
d EMPLOYEE_ID
t FIRST_NAME
t LAST_NAME
t EMAIL
t PHONE_NUMBER
t HIRE_DATE
t JOB_ID
d SALARY
t COMMISSION_PCT
t MANAGER_ID
d DEPARTMENT_ID

Target Datastore - ACC_EMPL

Indicators	Name	Mapping
n	EMPLOYEE_ID	EMPLOYEES.EMPLOYEE_ID
v	FIRST_NAME	EMPLOYEES.FIRST_NAME
v	LAST_NAME	EMPLOYEES.LAST_NAME
v	EMAIL	EMPLOYEES.EMAIL
v	PHONE_NUMBER	EMPLOYEES.PHONE_NUMBER
v	HIRE_DATE	EMPLOYEES.HIRE_DATE
v	JOB_ID	EMPLOYEES.JOB_ID
n	SALARY	EMPLOYEES.SALARY
v	COMMISSION_PCT	EMPLOYEES.COMMISSION_PCT
v	MANAGER_ID	EMPLOYEES.MANAGER_ID
n	DEPARTMENT_ID	EMPLOYEES.DEPARTMENT_ID

Note: - All the columns need to be mapped to staging only. Column mapped on Source is not supported and will throw error.

Select the LKM ACCESS to SQL_V1 and appropriate IKM depending on your Target Technology.



Source Set Properties

Name: SrcSet0

LKM Selector: LKM ACCESS to SQL V1

COMPONENT NAME:LKM ACC
COMPONENT VERSION: 1.0
AUTHOR: Kshitiz Devendra (C
COMPATIBILITY: ODI 10g anc

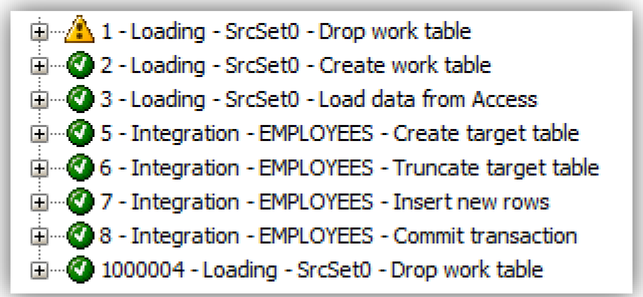
Description:

Options:

Name	Value
DELETE_TEMPORARY_OBJECTS	<default>:true

Save and Execute and ones the Execution is successful. Please look into the Target Table for Data.

Operator Session Execution of the above Interface

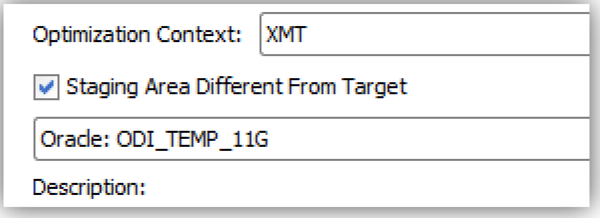


```
1 - Loading - SrcSet0 - Drop work table
2 - Loading - SrcSet0 - Create work table
3 - Loading - SrcSet0 - Load data from Access
5 - Integration - EMPLOYEES - Create target table
6 - Integration - EMPLOYEES - Truncate target table
7 - Integration - EMPLOYEES - Insert new rows
8 - Integration - EMPLOYEES - Commit transaction
1000004 - Loading - SrcSet0 - Drop work table
```

INTEGRATION INTO ACCESS (IKM)

This IKM is Multi Technology IKM and so does not create I$ table but load directly from C$ table / Source.

Enable the Option Staging Area Different from Target and select the appropriate Source Logical Schema.



Optimization Context: XMT

☑ Staging Area Different From Target

Oracle: ODI_TEMP_11G

Description:

The different options of the IKM are

CREATE_TARGET_TABLE: This option enables you to create a Target Table.

INSERT: Set it true if you want only to insert.

INSERT/UPDATE: Set it True if you want Insert/Update.

CREATE TARGET TABLE & INSERT ONLY

The IKM ACCESS Insert is a Multi Technology IKM .Create Target datastore with appropriate Column Name and Datatype.

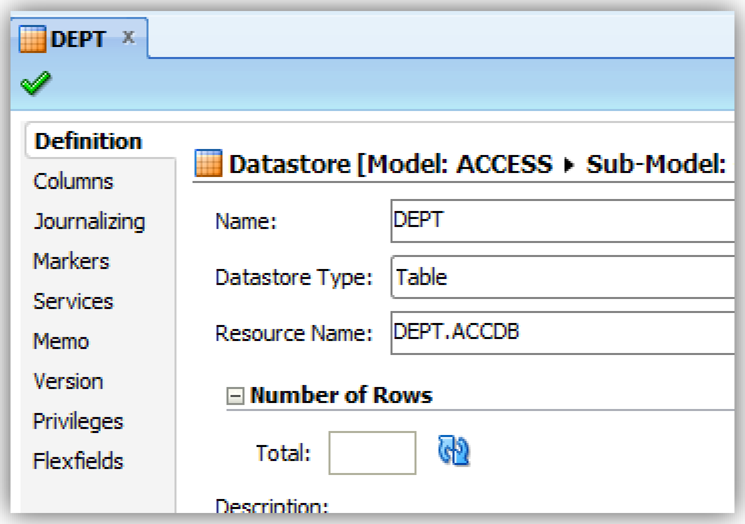
Data Store Name :   <<Table Name>>.

The Sheets created in ACCESS will have data Store Name.

Resource Name : <<File Name with Extensions>>

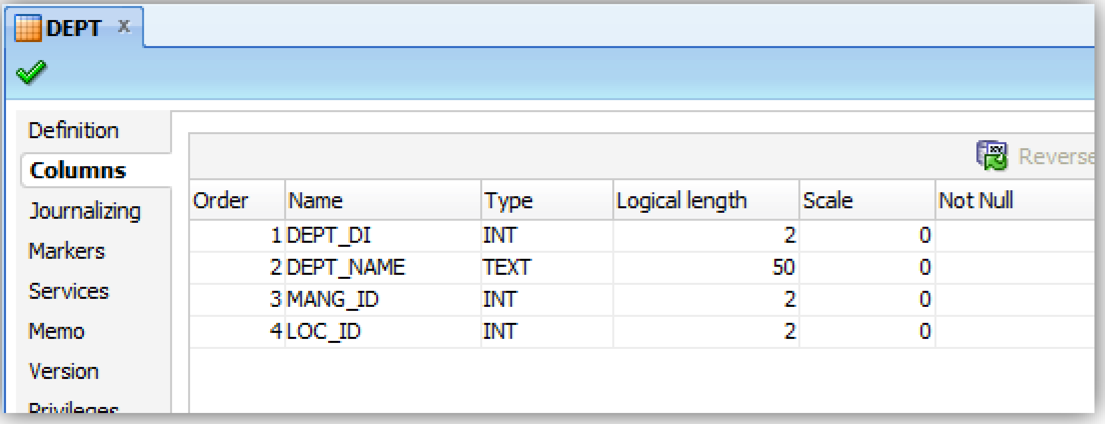
Make sure you provide the file with appropriate Extension mdb / accdb.

[Note: - Without Extension IKM will throw error.]



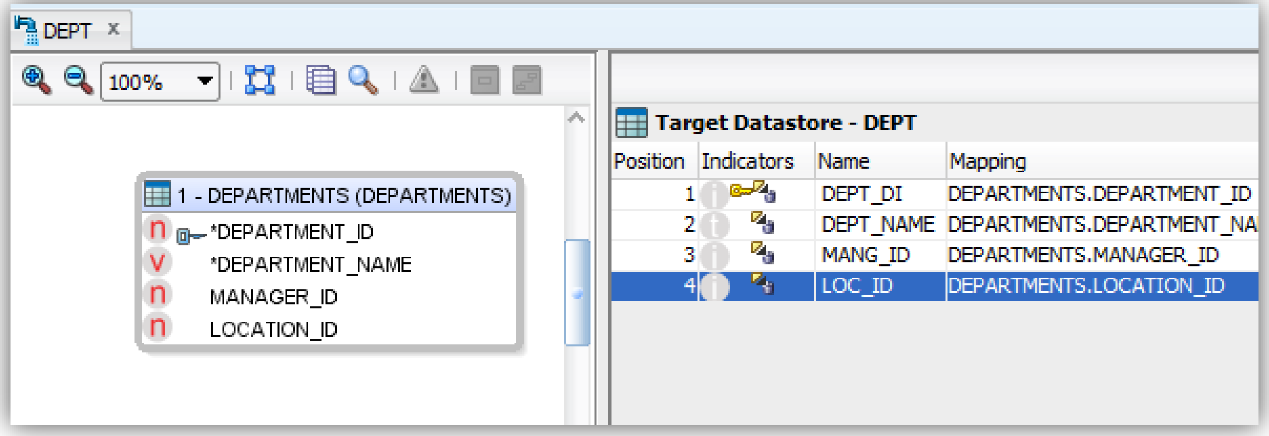
Name:	DEPT
Datastore Type:	Table
Resource Name:	DEPT.ACCDB

Select the appropriate Datatype, Length and Scale

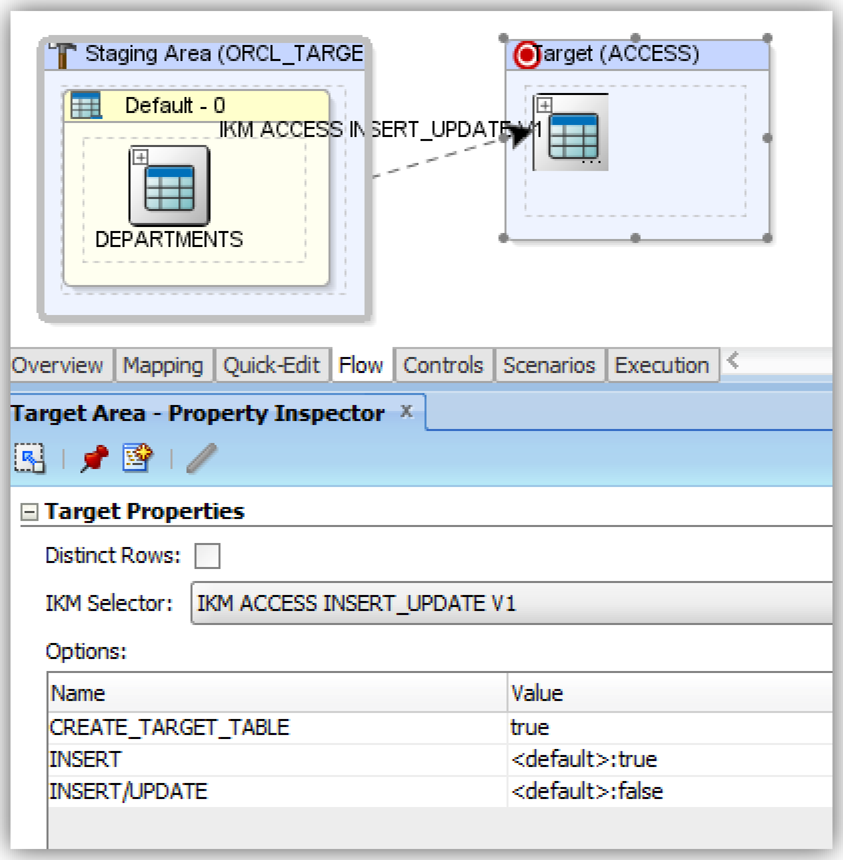


Order	Name	Type	Logical length	Scale	Not Null
1	DEPT_DI	INT	2	0	
2	DEPT_NAME	TEXT	50	0	
3	MANG_ID	INT	2	0	
4	LOC_ID	INT	2	0	

Map the Column accordingly.



Position	Indicators	Name	Mapping
1		DEPT_DI	DEPARTMENTS.DEPARTMENT_ID
2		DEPT_NAME	DEPARTMENTS.DEPARTMENT_NA
3		MANG_ID	DEPARTMENTS.MANAGER_ID
4		LOC_ID	DEPARTMENTS.LOCATION_ID



Target Properties

Distinct Rows: ☐

IKM Selector: IKM ACCESS INSERT_UPDATE V1

Options:

Name	Value
CREATE_TARGET_TABLE	true
INSERT	<default>:true
INSERT/UPDATE	<default>:false

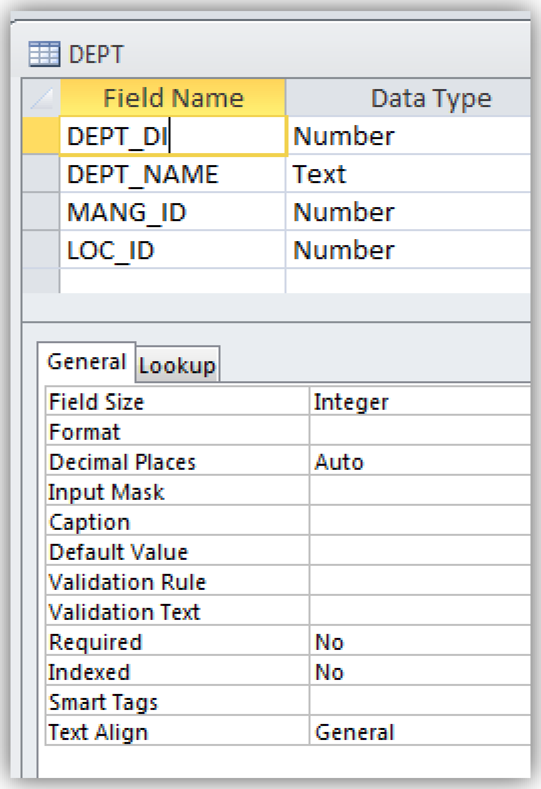
CREATE_TARGET_TABLE

This option will create table as defined in the Target datastore. If table already exist it won't replace so you will need to manually drop it for any column or datatype changes in data store.

INSERT

If you are just Inserting then use Insert as it uses a different algorithm for fast data load.

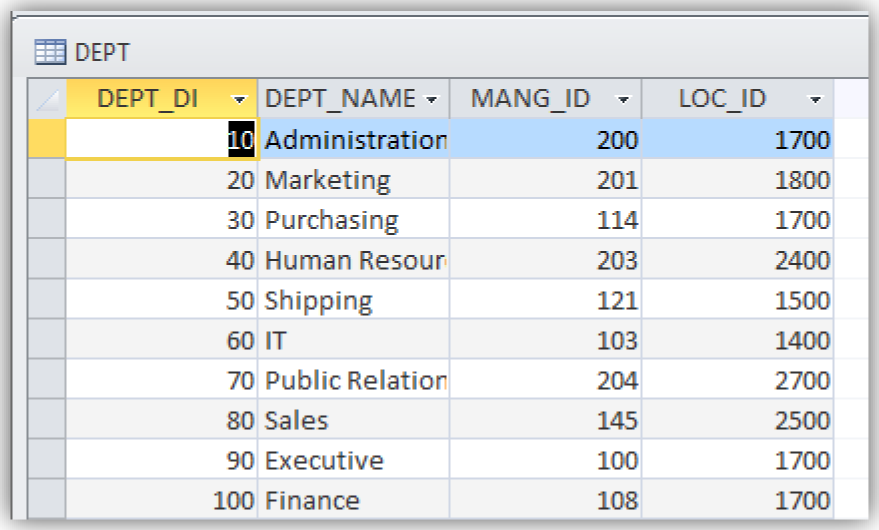
As you can see the Table is created with the structure defined in the ODI Datastore.


DEPT

Field Name	Data Type
DEPT_DI	Number
DEPT_NAME	Text
MANG_ID	Number
LOC_ID	Number

Field Size	Integer
Format	
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	No
Smart Tags	
Text Align	General

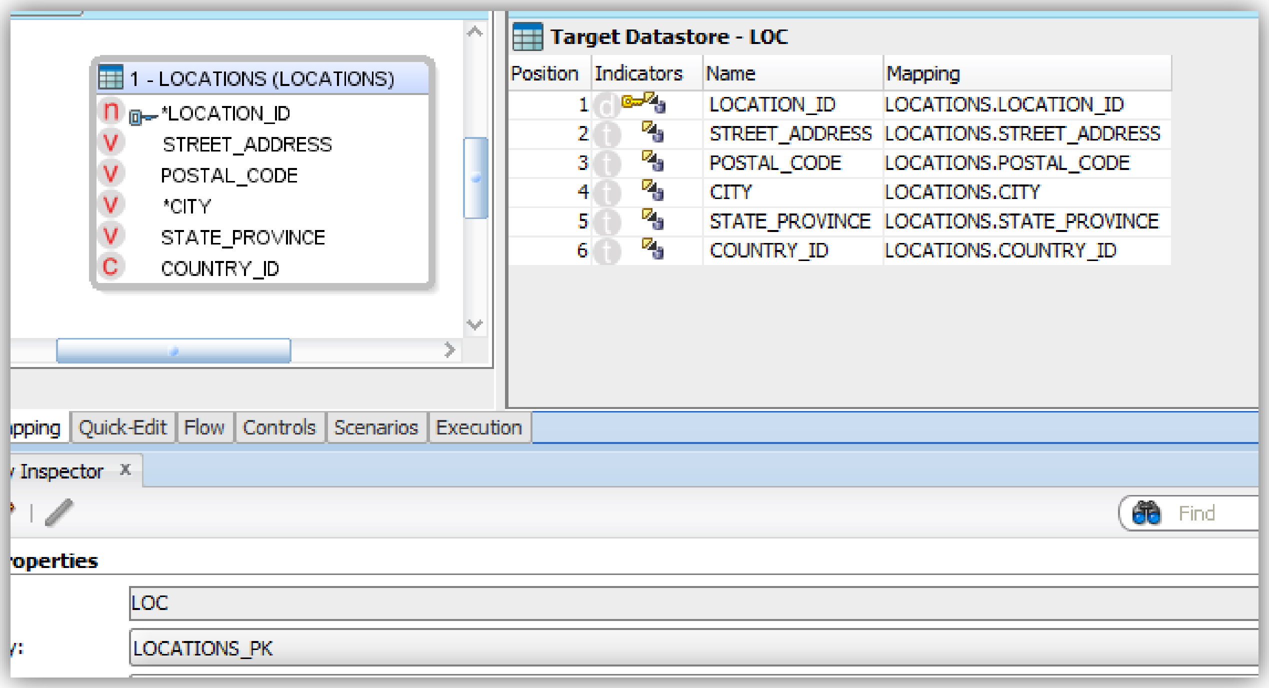
Sample Output Data



DEPT

DEPT_DI	DEPT_NAME	MANG_ID	LOC_ID
10	Administration	200	1700
20	Marketing	201	1800
30	Purchasing	114	1700
40	Human Resour	203	2400
50	Shipping	121	1500
60	IT	103	1400
70	Public Relation	204	2700
80	Sales	145	2500
90	Executive	100	1700
100	Finance	108	1700

INSERT/UPDATE



1 - LOCATIONS (LOCATIONS)

- n *LOCATION_ID
- V STREET_ADDRESS
- V POSTAL_CODE
- V *CITY
- V STATE_PROVINCE
- C COUNTRY_ID

Target Datastore - LOC

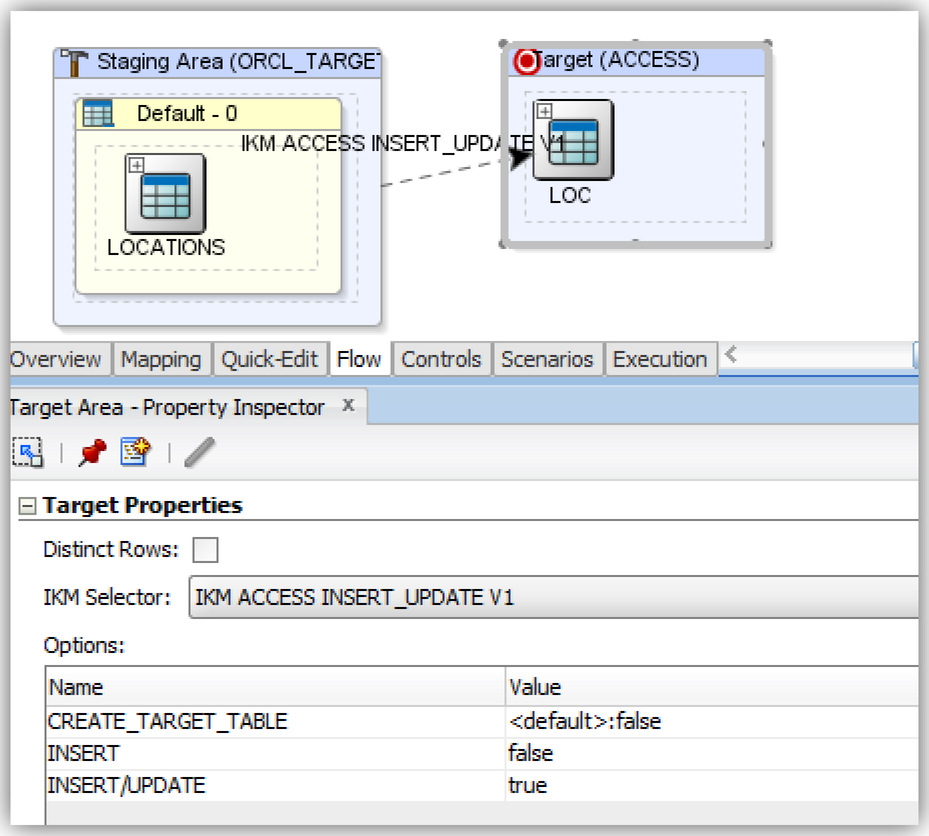
Position	Indicators	Name	Mapping
1	d	LOCATION_ID	LOCATIONS.LOCATION_ID
2	t	STREET_ADDRESS	LOCATIONS.STREET_ADDRESS
3	t	POSTAL_CODE	LOCATIONS.POSTAL_CODE
4	t	CITY	LOCATIONS.CITY
5	t	STATE_PROVINCE	LOCATIONS.STATE_PROVINCE
6	t	COUNTRY_ID	LOCATIONS.COUNTRY_ID

operties

LOC

y: LOCATIONS_PK

The update algorithm is configured in such a way that if a PK is present in access, driver will make use of the appropriate indexes for faster search and update.



Target Area - Property Inspector

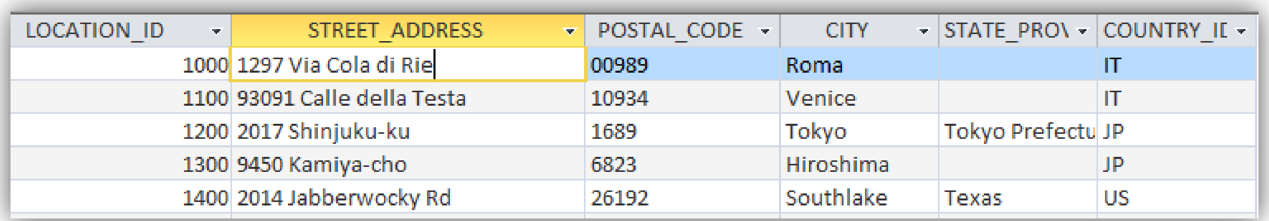
Target Properties

Distinct Rows: ☐

IKM Selector: IKM ACCESS INSERT_UPDATE V1

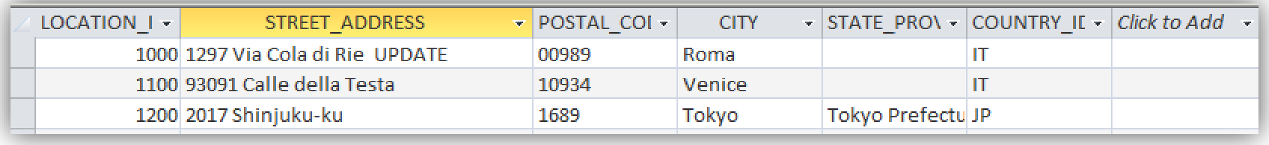
Options:

Name	Value
CREATE_TARGET_TABLE	<default>:false
INSERT	false
INSERT/UPDATE	true



LOCATION_ID	STREET_ADDRESS	POSTAL_CODE	CITY	STATE_PROV	COUNTRY_ID
1000	1297 Via Cola di Rie	00989	Roma		IT
1100	93091 Calle della Testa	10934	Venice		IT
1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefectu	JP
1300	9450 Kamiya-cho	6823	Hiroshima		JP
1400	2014 Jabberwocky Rd	26192	Southlake	Texas	US

Let's update the street address for location id 1000



LOCATION_I	STREET_ADDRESS	POSTAL_COI	CITY	STATE_PROV	COUNTRY_II	Click to Add
1000	1297 Via Cola di Rie UPDATE	00989	Roma		IT	
1100	93091 Calle della Testa	10934	Venice		IT	
1200	2017 Shinjuku-ku	1689	Tokyo	Tokyo Prefectu	JP	

DATATYPE

- BOOLEAN
- BYTE
- CURRENCY
- DATETIME
- DECIMAL
- DOUBLE
- FLOAT
- GUID / REPLICATION_ID (presently not supported)
- INT
- LONG
- MEMO
- OLE
- TEXT

API

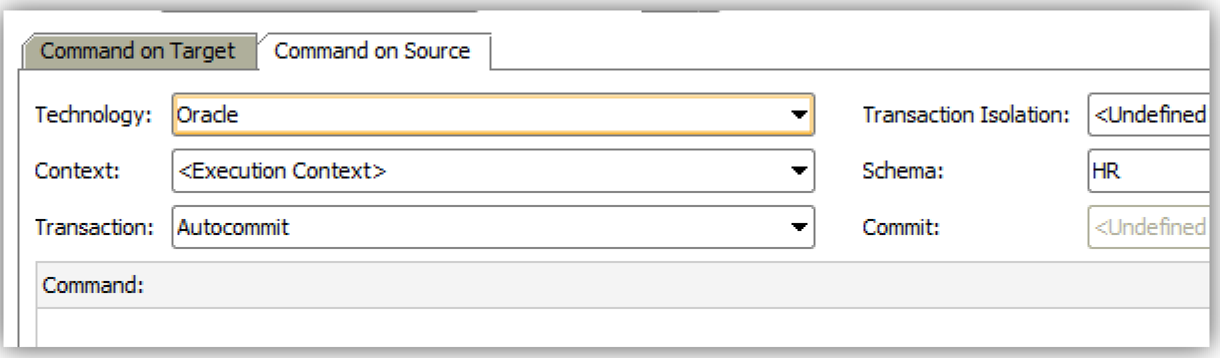
There are two API created that can be used in the ODI Procedure for copying data from Query and File, without creating any Interface - CopyTable & CopyFile

COPYTABLE

```
CopyTable (String Dir, String AccessFile, String tableName,
Connection conn, String SQL)
```

This method will allow you to copy Table from any SQL DB to Access.

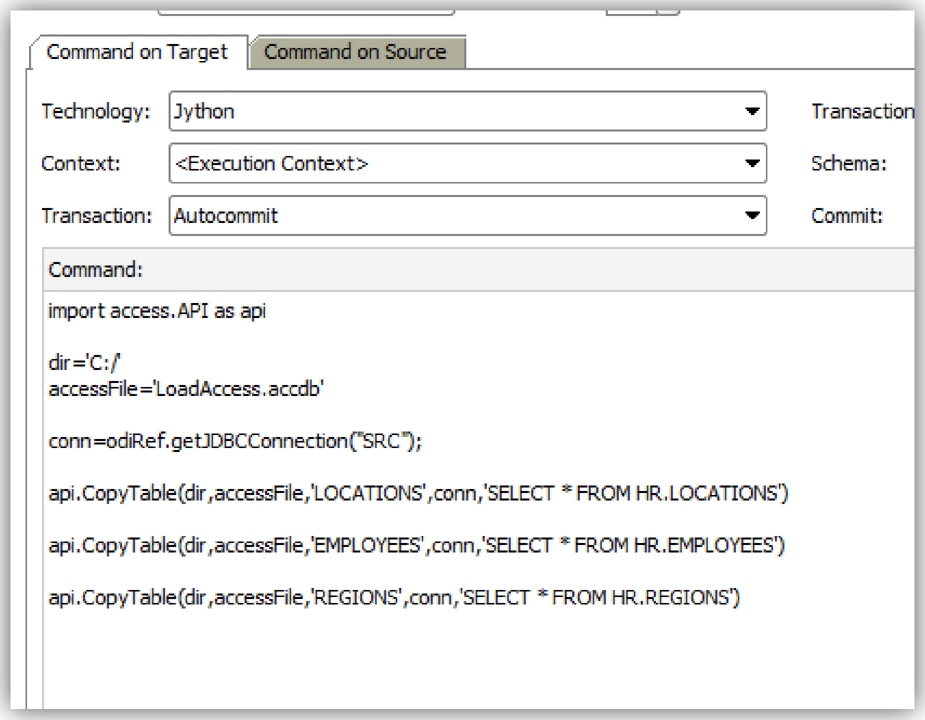
Command on Source - please specify the Technology and Schema.



Technology: Oracle
Context: <Execution Context>
Transaction: Autocommit
Transaction Isolation: <Undefined
Schema: HR
Commit: <Undefined
Command:

On the Command on Target specify the parameters for the Method with Jython as the technology. A sample screenshot is shown below.
You can load multiple tables from the same schema into the same/different ACCESS DBs. Depending on the datatype of the Source the appropriate datatype is calculated and set.

[Note: - Select * will fetch all the column, for selective columns please provide the column name in the Select statement]



Command on Target | Command on Source

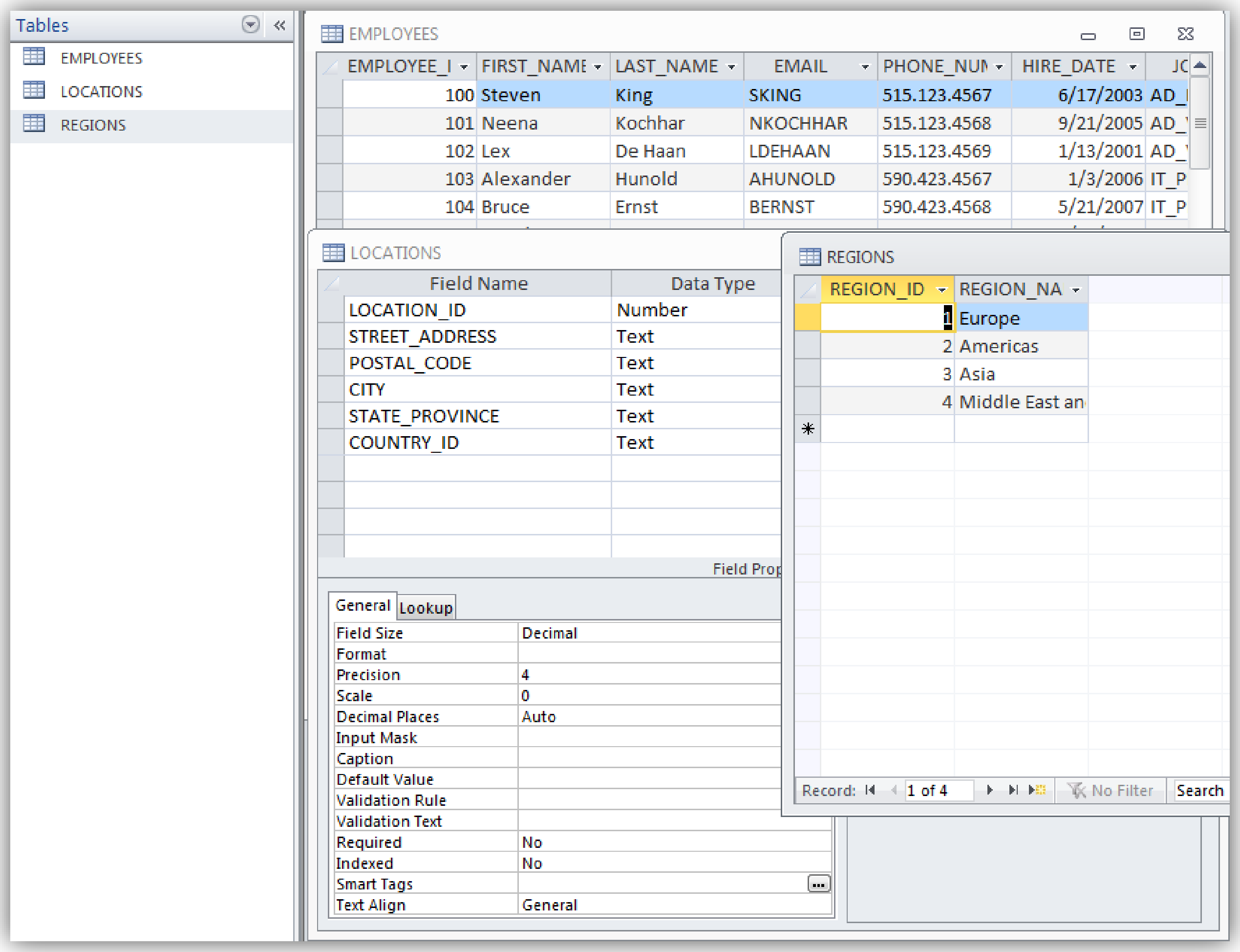
Technology: Jython
Context: <Execution Context>
Transaction: Autocommit
Transaction
Schema:
Commit:

Command:

```
import access.API as api

dir='C:/'
accessFile='LoadAccess.accdb'

conn=odiRef.getJDBCConnection("SRC");

api.CopyTable(dir,accessFile,'LOCATIONS',conn,'SELECT * FROM HR.LOCATIONS')

api.CopyTable(dir,accessFile,'EMPLOYEES',conn,'SELECT * FROM HR.EMPLOYEES')

api.CopyTable(dir,accessFile,'REGIONS',conn,'SELECT * FROM HR.REGIONS')
```

```
import access.API as api

dir='C:/'
accessFile='LoadAccess.accdb'

conn=odiRef.getJDBCConnection("SRC");

api.CopyTable(dir,accessFile,'LOCATIONS',conn,'SELECT * FROM
HR.LOCATIONS')

api.CopyTable(dir,accessFile,'EMPLOYEES',conn,'SELECT * FROM
HR.EMPLOYEES')

api.CopyTable(dir,accessFile,'REGIONS',conn,'SELECT * FROM
HR.REGIONS')
```

Sample Output of the above Procedure execution



Tables

- EMPLOYEES
- LOCATIONS
- REGIONS

EMPLOYEES

EMPLOYEE_I	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NUI	HIRE_DATE	JC
100	Steven	King	SKING	515.123.4567	6/17/2003	AD_
101	Neena	Kochhar	NKOCHHAR	515.123.4568	9/21/2005	AD_
102	Lex	De Haan	LDEHAAN	515.123.4569	1/13/2001	AD_
103	Alexander	Hunold	AHUNOLD	590.423.4567	1/3/2006	IT_P
104	Bruce	Ernst	BERNST	590.423.4568	5/21/2007	IT_P

LOCATIONS

Field Name	Data Type
LOCATION_ID	Number
STREET_ADDRESS	Text
POSTAL_CODE	Text
CITY	Text
STATE_PROVINCE	Text
COUNTRY_ID	Text

Field Prop

General | Lookup

Field Size	Decimal
Format	
Precision	4
Scale	0
Decimal Places	Auto
Input Mask	
Caption	
Default Value	
Validation Rule	
Validation Text	
Required	No
Indexed	No
Smart Tags	
Text Align	General

REGIONS

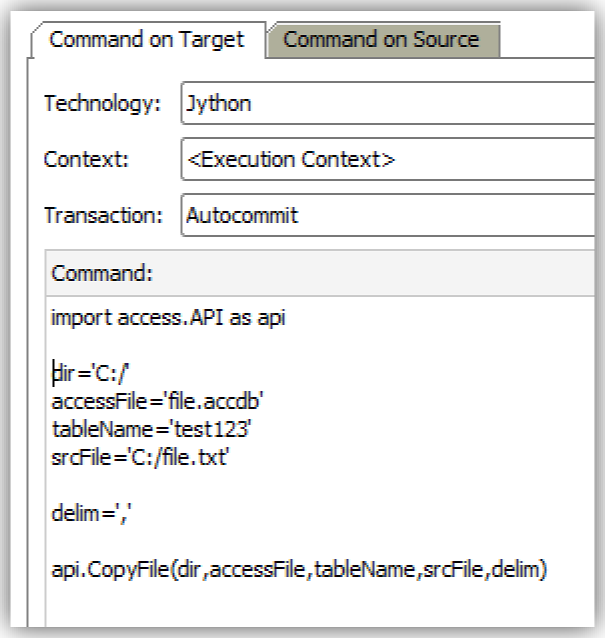
REGION_ID	REGION_NA
1	Europe
2	Americas
3	Asia
4	Middle East an

Record: 1 of 4 No Filter Search

COPYFILE

```
CopyFile (String Dir, String AccessFile, String tableName, String
srcFile, String delim)
```

Command on Target –Jython. It's a very simple method to load data from delimited File to Access. The File is read using a Buffered Reader so reading should be comparatively faster. Also the first line of the File is taken as Column Name and datatype is usually TEXT.



Technology: Jython

Context: <Execution Context>

Transaction: Autocommit

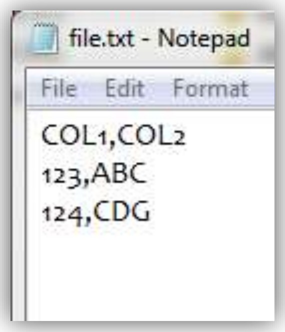
Command:

```
import access.API as api

dir='C:/'
accessFile='file.accdb'
tableName='test123'
srcFile='C:/file.txt'

delim=','

api.CopyFile(dir,accessFile,tableName,srcFile,delim)
```

[Note: - There is no option to pick selective column of the File .In order to do so please load via Interface]

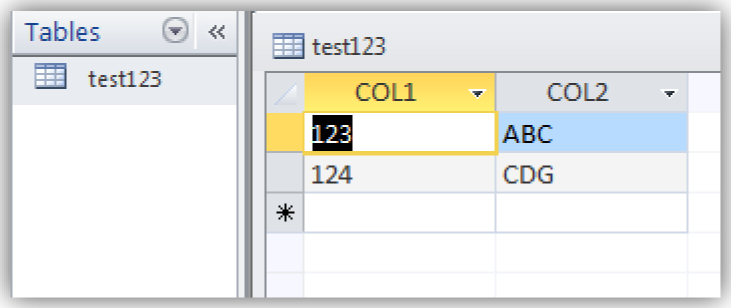
```
import access.API as api

dir='C:/'
accessFile='file.accdb'
tableName='test123'
srcFile='C:/file.txt'

delim=','

api.CopyFile(dir,accessFile,tableName,srcFile,delim)
```

Source File



```
COL1,COL2
123,ABC
124,CDG
```

Target ACCESS DB



COL1	COL2
123	ABC
124	CDG

COMMON ERROR/ISSUES

We have documented some of the common error/issues which can occur due to invalid entry/options in KM options or Excel File.

FileNotFoundException: - This exception will occur when you provided an invalid Access DB or Access File without correct Extension.

Data does not get loaded from Excel (LKM):- While using LKM Access to SQL, if your interface is successful yet you don't see any data in target, it means you have set the column mapped to Source. Please set the appropriate column mapping to Staging and try again

CONTACT US

We hope that this documentation has helped in the initial steps of using and understanding the ODI JDBC Access driver and KM's.

In case if you still have issues or question please feel free to contact us at www.odiexperts.com.