Denodo DynamoDB Custom Wrapper

Revision 20220804

Denodo Inc.: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA www.denodo.com

CONTENTS

1 INTRODUCTION

The Denodo DynamoDB Custom Wrapper is a Virtual DataPort custom wrapper for querying Amazon DynamoDB tables.

1.1 WHAT IS DYNAMODB?

Amazon DynamoDB is a NoSQL database service offered by AWS. It has a key-value design.
It models the data into tables, which contain items, and these items have attributes. Items are schema-less, but when a table is created it is mandatory to specify a primary key.

DynamoDB is designed for high-performance at any scale.

1.2 ARCHITECTURE AND FEATURES

The Denodo DynamoDB custom wrapper allows us to create base views on DynamoDB tables and execute SQL queries on those collections.

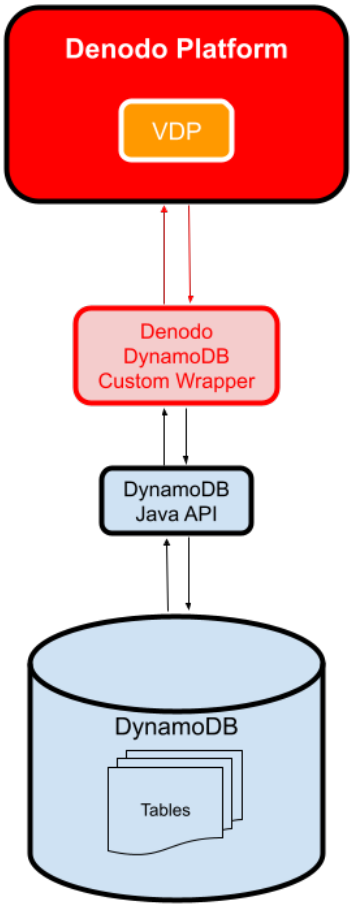
This component uses the Amazon DynamoDB module in the AWS Java SDK for communicating with the Amazon DynamoDB Service. The wrapper is read-only. It supports authentication via Access Key ID / Secret Access Key, plus assumed IAM roles and region. It delegates filtering and projections to the amount made possible by the binary DynamoDB Java APIs.

DynamoDB offers two operations for exploring and retrieving data from a table: `Scan` and `Query`. `Query` finds items based on indexed values while `Scan` looks for the desired results by scanning every item of the table (returns all the items by default). For performance reasons, the `Query` operation is a better option than `Scan` when queries include a condition on indexed fields (which are mandatory when `Query` is used).

This distribution includes two wrappers:

- **`com.denodo.connect.dynamodb.DynamoDBScanWrapper`** which uses the `Scan` operation.
- **`com.denodo.connect.dynamodb.DynamoDBQueryWrapper`** which uses the `Query` operation and requires the specification, at configuration time, of an index (primary or secondary) to be used. This wrapper will require all queries to include a condition on the indexed fields (on the partition key and optionally also on the sort key).

(Note there is a third implementation called `com.denodo.connect.dynamodb.DynamoDBWrapper`, which is totally equivalent to `DynamoDBScanWrapper` and should be considered deprecated).



Denodo DynamoDB Custom Wrapper architecture

This is a brief summary of the wrapper's current features:

- Access Key ID / Secret Access Key authentication and IAM roles are supported.
- Allows the explicit specification of base view schema in order to adapt schema-less DynamoDB to relational VDP. The following rules apply for matching table items to the schema of the base view:
    - Extra fields in returned items are ignored.
    - Lacking fields in returned items are returned as `null`.
    - At base view creation time, data types can be specified from constants in `java.sql.Types`, which correspond to SQL standard types.
- Alternatively the schema could be defined using an introspection query. **All items** retrieved by this query are analyzed to reveal their fields and build the base view schema. For this reason the query should retrieve a significant sample of the collection we are interested in.

    - Since common fields may hold different types of data the resulting structure is the highest common denominator between all the fields with the same name. In case of incompatible fields --such as `integer` and `maps`-- VDP interprets them as of type `text`.

- Field projections are delegated to DynamoDB.
- Some query conditions are delegated to DynamoDB.
    - AND and OR conditions.
    - Operators: `=`, `<>`, `<`, `>`, `<=`, `>=`, `IN`, `IS NULL`, `IS NOT NULL` and `BETWEEN`.
        - Note the `IS NULL` operator gets the documents where the specified field has a `null` value or does not exist.

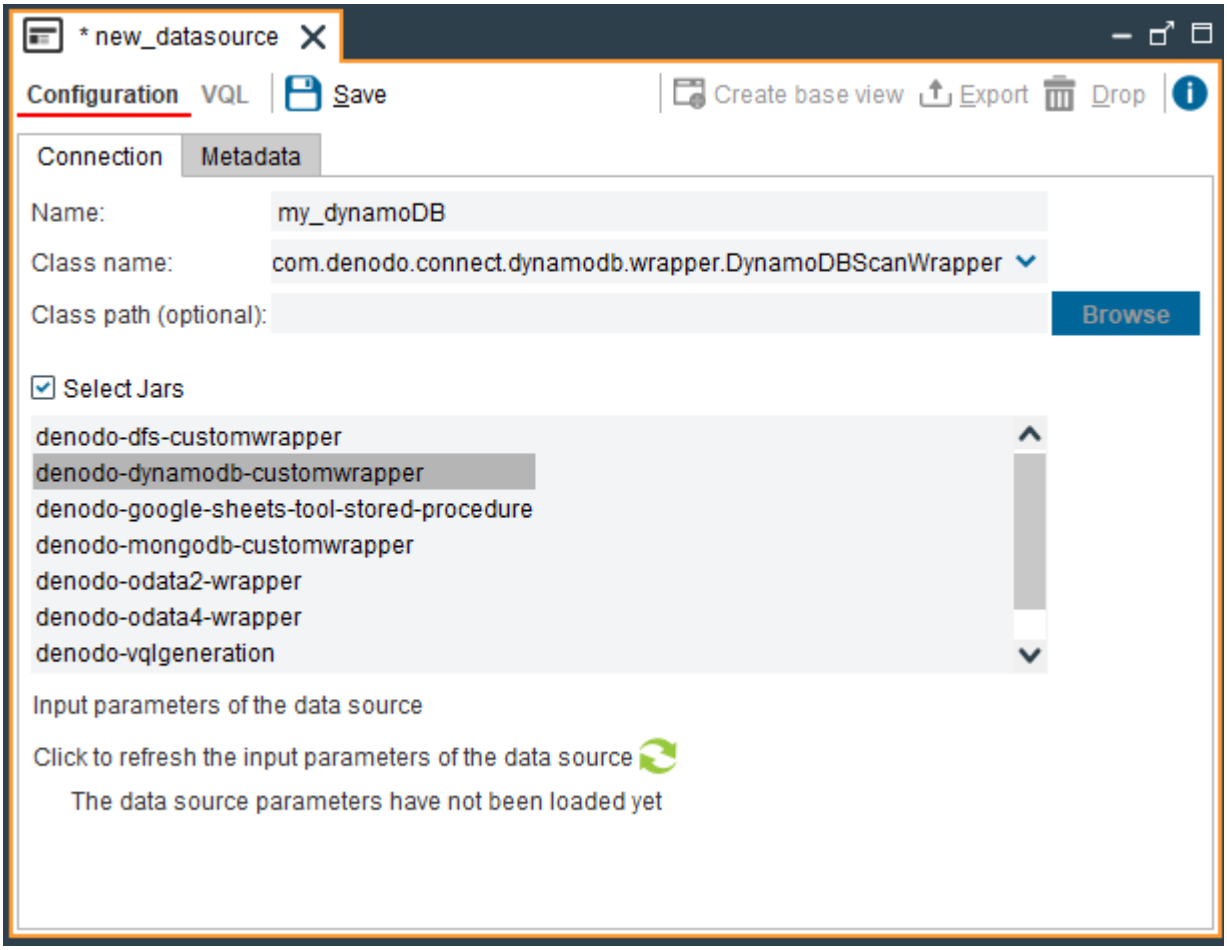
2 USAGE

2.1 IMPORTING THE CUSTOM WRAPPER INTO VDP

In order to use the Denodo DynamoDB Custom Wrapper in VDP, first it is necessary to install its JAR file into the Denodo VDP server. For this, in the VDP administration tool, go to the `File > Extensions` menu option and install the `jar-with-dependencies` file that is included in the distribution of the Denodo DynamoDB Custom Wrapper.

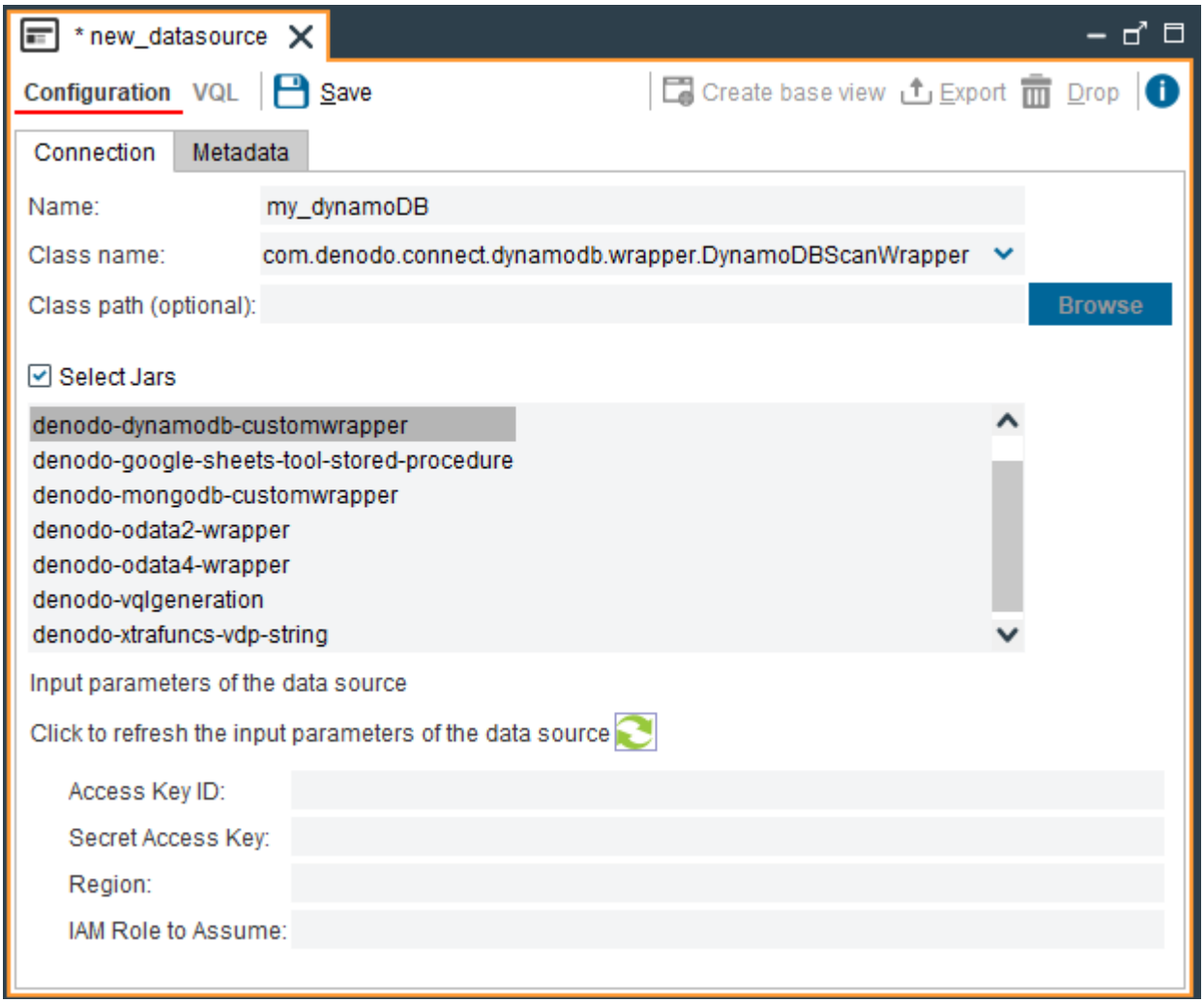
2.2 CREATING A DYNAMODB DATA SOURCE

- Go to `New > Data Source > Custom`
- Select the `denodo-dynamodb-customwrapper` jar
- Specify one of the two possible wrappers:
  - **`com.denodo.connect.dynamodb.wrapper.DynamoDBScanWrapper`** (Scan operation in DynamoDB)
  - **`com.denodo.connect.dynamodb.wrapper.DynamoDBQueryWrapper`** (`Query` operation in DynamoDB)
  - *(note `com.denodo.connect.dynamodb.wrapper.DynamoDBWrapper` is deprecated)*
- Click OK.
- Click to refresh the input parameters of the data source.


* new_datasource
Configuration
VQL
Save
Create base view
Export
Drop
Connection
Metadata
Name:
my_dynamoDB
Class name:
com.denodo.connect.dynamodb.wrapper.DynamoDBScanWrapper
Class path (optional):
Browse
Select Jars
denodo-dfs-customwrapper
denodo-dynamodb-customwrapper
denodo-google-sheets-tool-stored-procedure
denodo-mongodb-customwrapper
denodo-odata2-wrapper
denodo-odata4-wrapper
denodo-vqlgeneration
Input parameters of the data source
Click to refresh the input parameters of the data source
The data source parameters have not been loaded yet

2.2.1 Data source parameters

2.2.1.1 DynamoDBScanWrapper

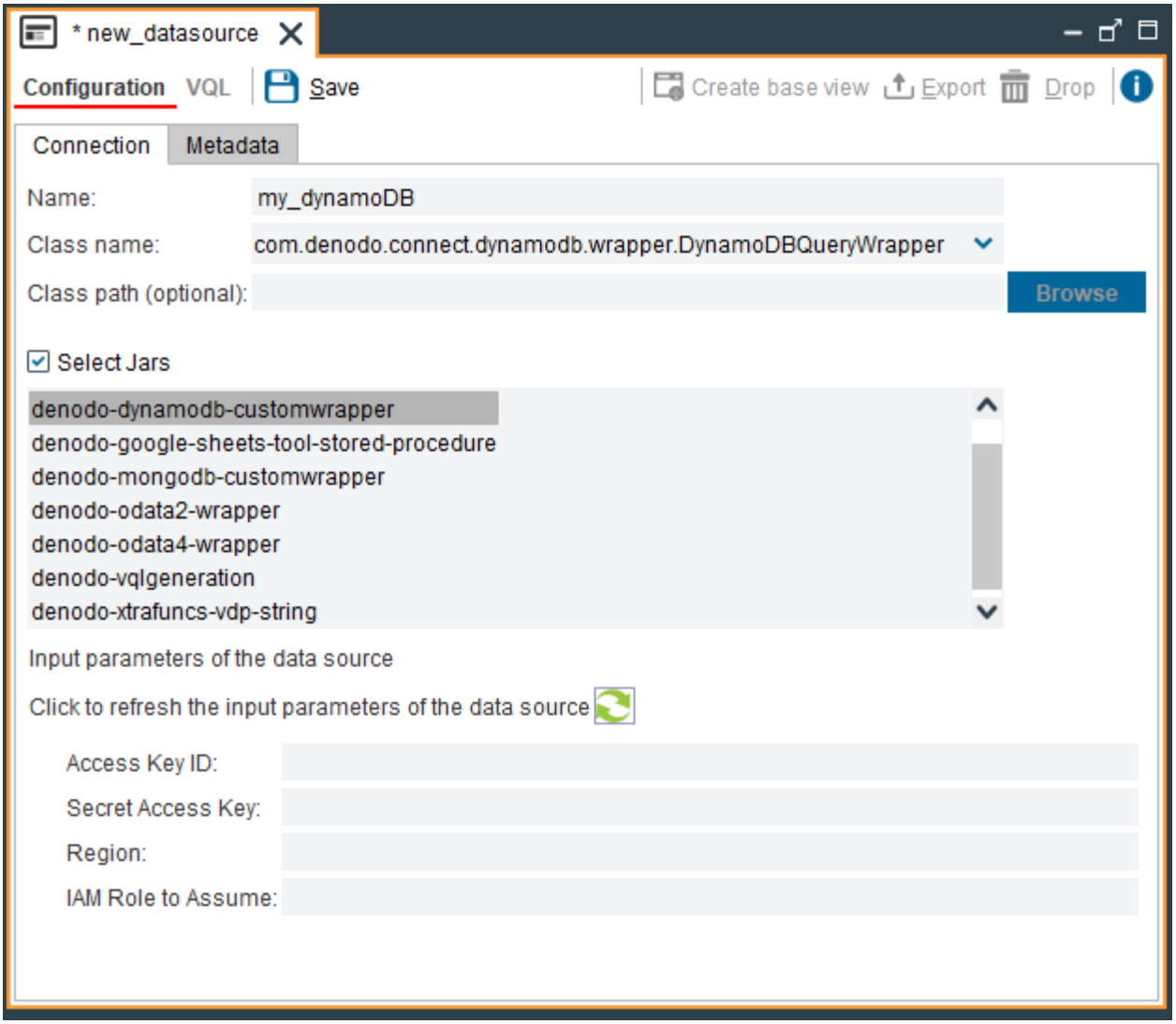


At the data source level, it is necessary to fill in the parameters related with authentication in DynamoDB.

- **Access Key Id:** AWS access key ID.
- **Secret Access Key**: Secret for the access key.
- **Region:** Region where the tables to be accessed are available.
- **IAM Role to Assume**: IAM role to be assumed by the AWS client in order to access.

None of these fields are mandatory.

2.2.1.2 DynamoDBQueryWrapper



At the data source level, it is necessary to fill in the parameters related with authentication in DynamoDB.

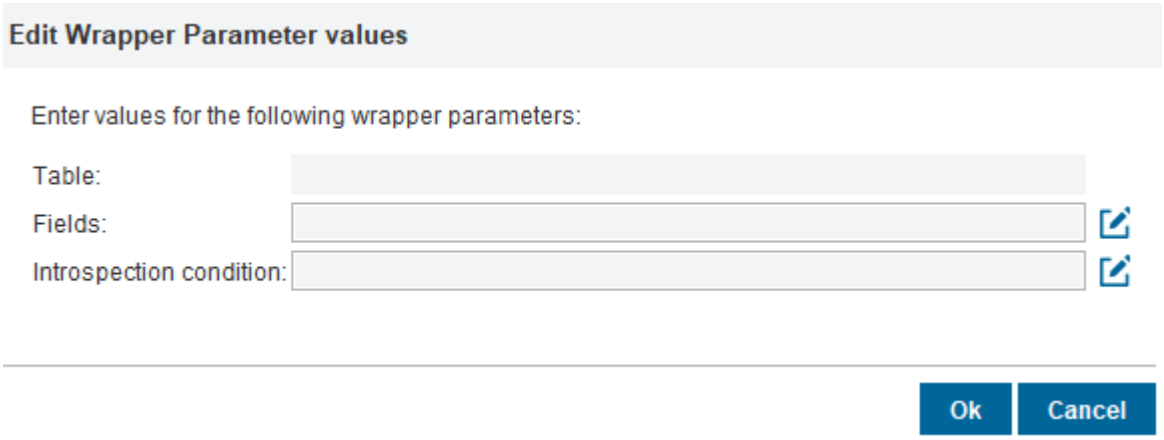
- **Access Key Id:** AWS access key ID.
- **Secret Access Key**: Secret for the access key.
- **Region:** Region where the tables to be accessed are available.
- **IAM Role to Assume**: IAM role to be assumed by the AWS client in order to access.

None of these fields are mandatory.

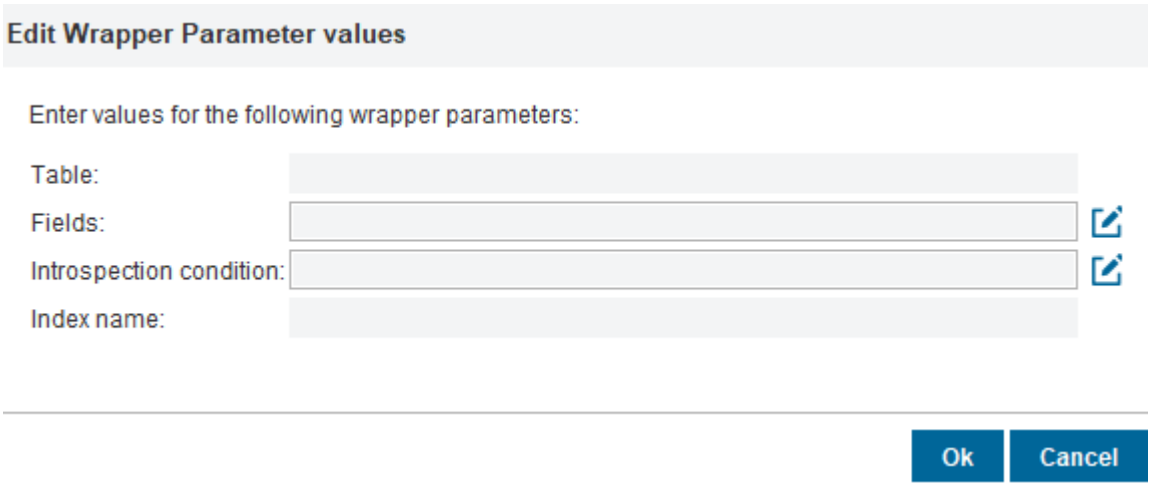
2.3 CREATING BASE VIEWS

Once the custom wrapper has been registered, click on `Create a base view`.

This looks like this in `DynamoDBScanWrapper`:



And it looks like this in `DynamoDBQueryWrapper`:



2.3.1 Base view parameters

- **Table:** DynamoDB table that will be accessed by this base view.

There are also two parameters that are **mutually exclusive**:

- **Fields:** The fields we would like to import as columns in VDP. We must keep the syntax
`field1:type1[,field2:type2,...]`.
Type should be one of the constants in `java.sql.Types` (note these are SQL standard types). See a specific section below to learn more about the allowed syntax.

- **Introspection Condition**: This field allows the specification of a filter to be applied on a query on the DynamoDB table, so that the results of this query are used for inferring the base view schema by analyzing their fields and data types. The results of this filtered query must be representative enough for all the intended fields to be correctly modeled as a result of its execution. And this condition isn't used in the execution of the base view, only in the creation of the base view schema.
  NOTE that if this filter allows for the return of too many results, the creation of the base view can take a long time. In addition, it is recommended to check the structure and types of the new view's fields inferred from the introspection.
  The syntax of the "introspection condition" only supports simple conditions. The operators allowed are `=`, `<`, `<=`, `>`, `>=` , `is null`, `is not null`, `contains` and `between`. The operator has to be between blank spaces.
  The syntax for between is: `id between 5 and 7`.
  In the right operand the string has to be between double quotes: `name = "Paul"`.
  For timestamps, double-quoted values in ISO8601 format need to be used.

- **Index name** (only for `DynamoDBQueryWrapper`): if an index is not specified or its value is "Primary", the table key partition and the table sort key will be used as a query index. Alternatively, the name of a secondary index can also be specified: in such a case the wrapper will consider the key partition of that index as mandatory in all queries (using an equality operator), and will also optionally support the use of a condition on the index's sort key..

2.3.2 Schema definition

2.3.2.1 Fields

When using the `Fields` parameter, we will need to specify the fields we would like to import as columns in VDP, using this syntax:

`field1:type1[,field2:type2,...]`.

Type should be one of the constants in `java.sql.Types` (note these are SQL standard types).

- Default is VARCHAR.
- If items in the same collection specify different types for fields with the same name, using VARCHAR (`text` in VDP) for that column will automatically perform the required conversions to show data from those items.

```
Year: integer,
Title: varchar,
Info:\{ directors: varchar,
        release_date: timestamp,
        genres:varchar,
```

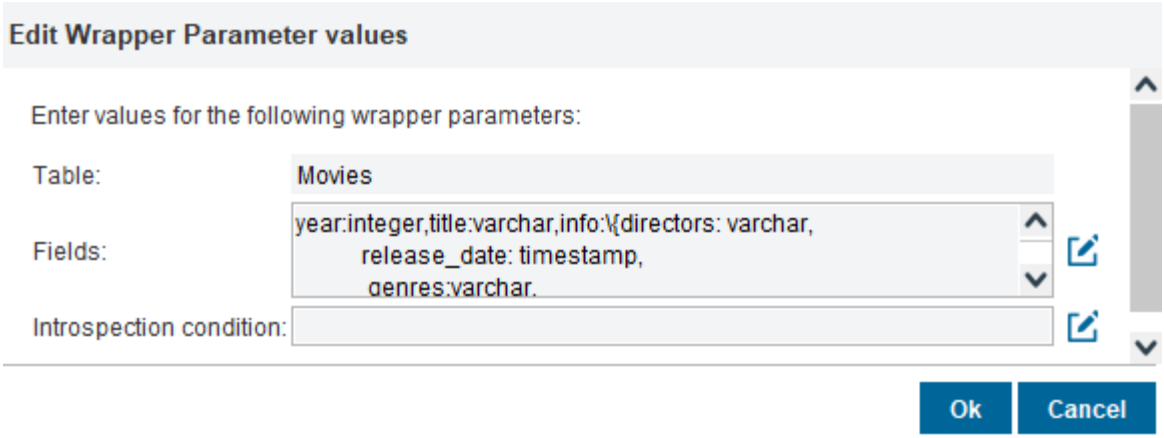
```
            image_url: varchar,
            plot: varchar,
            rank: integer,
            running_time_secs: integer,
            Actors: array(varchar)
    \}
```

See how complex structures are specified using braces (`{...}`), which need to be escaped, and arrays are specified using the `array` keyword and specifying the contents of the array between parentheses.

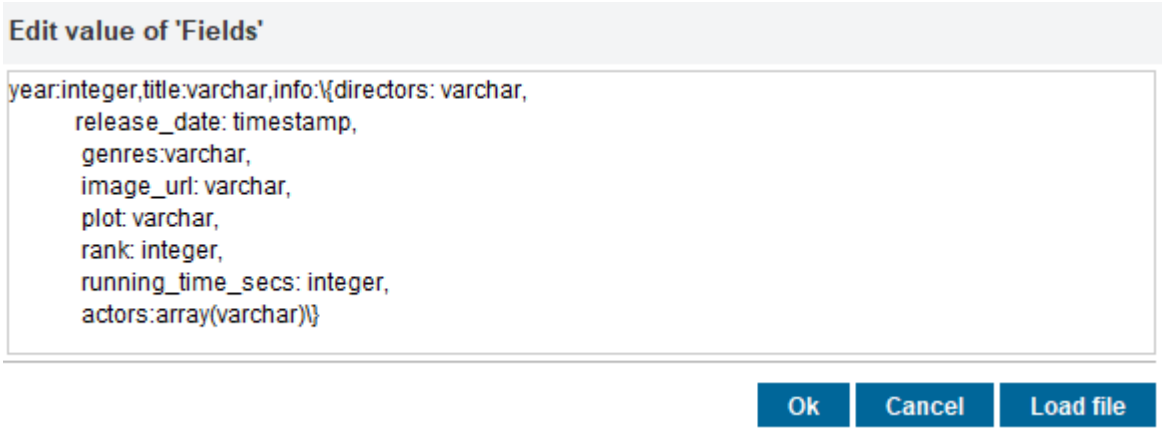
Also field names can be surrounded by single or double quotes (e.g. `"phone number"`) if they contain white spaces or non-alphanumeric chars.

2.3.2.1.1 Example

In the following example we import the table Movies from Dynamodb to VDP using the parameter `Fields`.

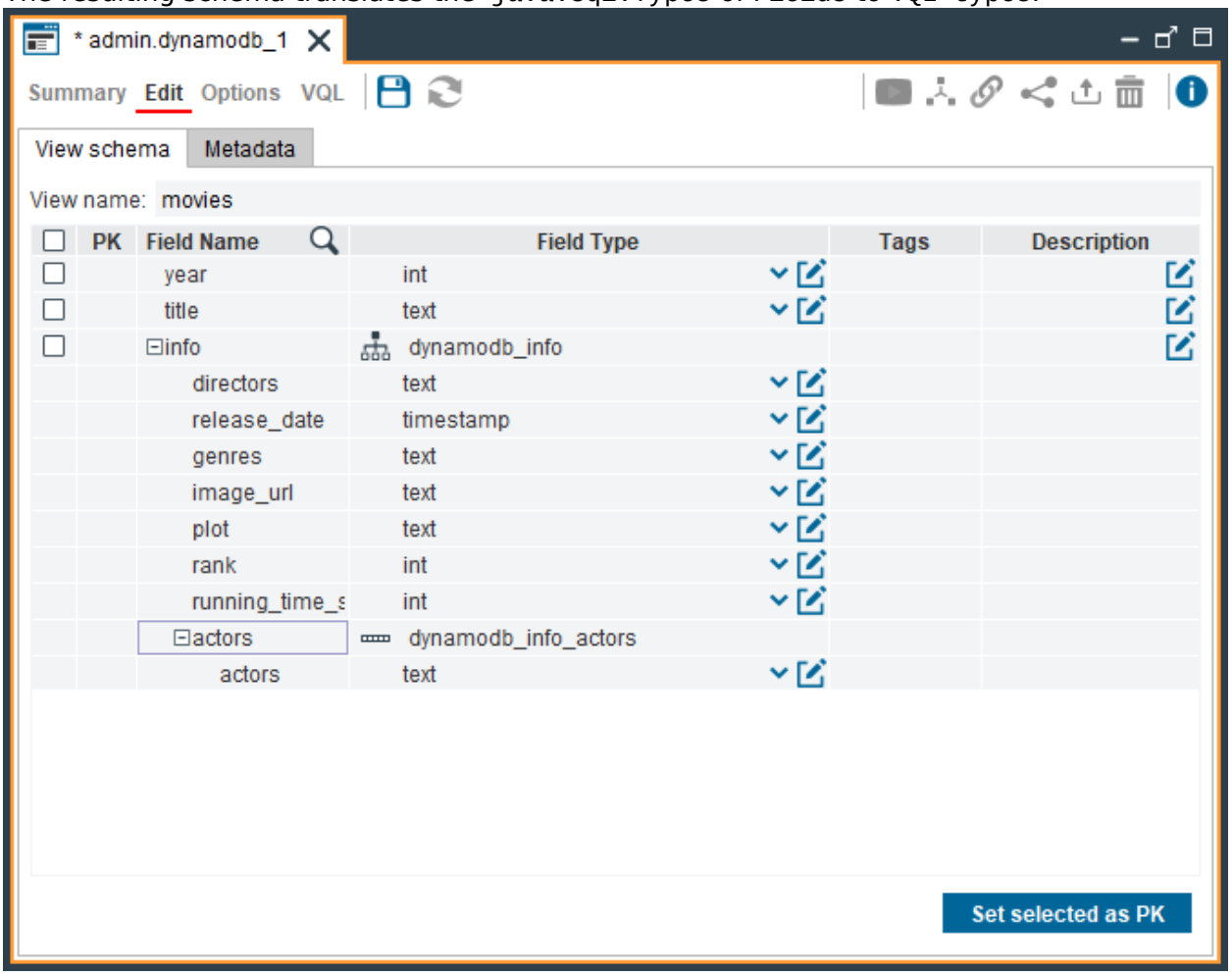


2.3.2.2



Example of `Fields`

The resulting schema translates the `java.sql.Types` of `Fields` to `VQL types`.



DynamoDB Base View

2.3.2.3

2.3.2.4 Introspection Condition

The user has the option not to explicitly define the schema of the base view, using the `Condition Introspection` parameter. This parameter has a simple condition to filter the items of a DynamoDB table. Based on the results of the query the custom wrapper builds an schema, when the same field appears several times, the type of this field will be the more generic.

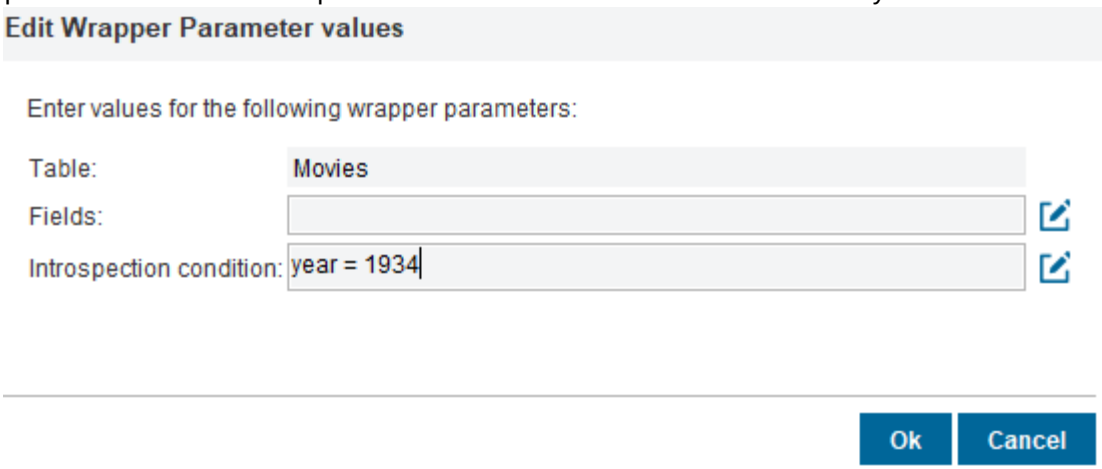
The condition only can be on simple attributes, it isn't possible to filter on arrays or structs. The syntax of this field is: `<operand> <operator> <value>`. The allowed operators are:
`=, <>, <, <=, >, >=, CONTAINS` and `BETWEEN`. The operator has to be between blank spaces. The `CONTAINS` operator only supports String type.

The values of the string and date types have to be between double quotes, and in the case of date type have to be represented according to the ISO8601 standard format.

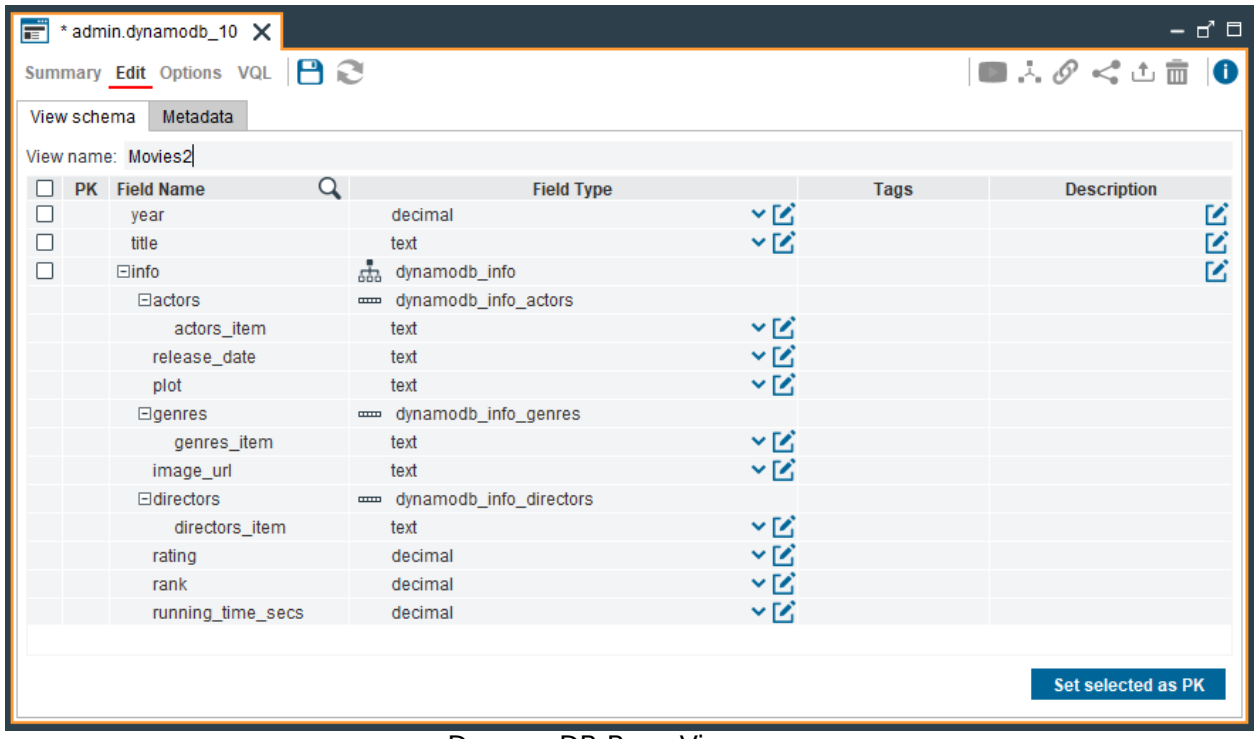
```
Id                              =                               5
Name = "Bryan"
Year            BETWEEN         1920            and             1930
```

2.3.2.4.1 Example

In the following example we import the table Movies from Dynamodb to VDP using the parameter `Introspection condition` with `year = 1934`



Example using Introspection condition



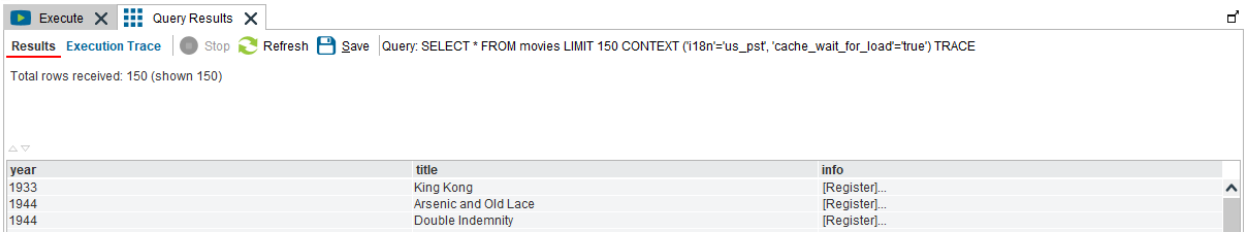
DynamoDB Base View

2.3.3 Execution

2.3.3.1 DynamoDBScanWrapper

The `DynamoDBScanWrapper` implementation does not pose any specific limitations on the conditions or projections of queries. As mentioned, this wrapper delegates the data retrieval operation to DynamoDB as a `Scan` call.

2.3.3.2



2.3.3.3 DynamoDBQueryWrapper

In `DynamoDBQueryWrapper` implementation uses a `Query` call in DynamoDB, and establishes a series of limitations on queries so that they adapt to the requirements of `Query` calls.

Specifically, a condition on the configured index's partition key fields has to be included in the `WHERE` clause of the query performed on the base view, and this condition needs to use the equality operator (=). A condition on the sort key of the index can also be specified, and this sort condition can use these operators: =, < , <=, > , >= and `between`.

Only one partition condition and one sort condition can be specified in the query. These index and sort conditions need to be specified at the first level of the `WHERE` tree of conditions, and they need to be joined with `AND` between them and also with the root of the rest of the condition tree on other fields.

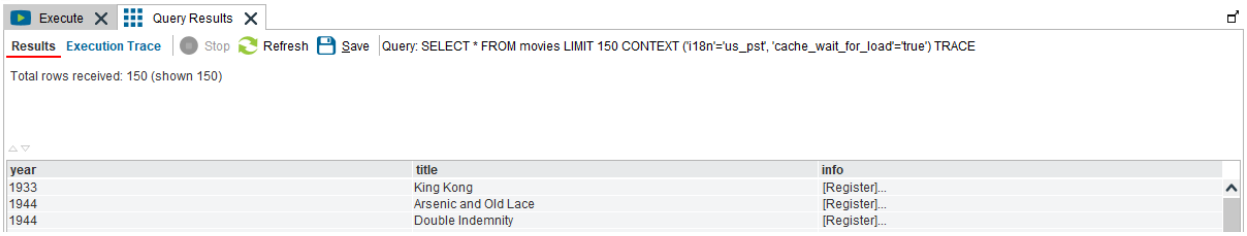
Conditions on other fields in the base view can be added to the `WHERE` clause without restrictions.

Example:

```
SELECT                                                              *
FROM                                                            table
WHERE              keyPartition              =              'keypartition'
                 AND        sortPartition        <        'sortpartition'
  AND otherFields <> 'otherFields'
```

Note that `keyPartition` is mandatory and `sortPartition` and the rest of the fields participating in the query conditions are optional.

2.3.3.4



DynamoDB Base View execution

3 LIMITATIONS

- Create/Update operations are not supported.
- The `ORDER BY` clause isn't delegated to DynamoDB.
- The `LIKE` operator isn't delegated to DynamoDB.
- The custom wrapper cannot read attributes of complex types of DynamoDB such as `List` and `Map` if the attributes have names including capital letters.