Denodo Hibernate Dialect User Manual

Revision 20190104

Denodo North America & APAC: 525 University Avenue, Suite 31, Palo Alto, CA 94301. USA
Denodo Iberia & Latino América: Montalbán 5, 28014 Madrid, Spain
Denodo EMEA: 21st Floor, Portland House, Bressenden Place, London SW1E 5RS. UK
Denodo DACH: Karlstraße 10, 80333 München. Germany
www.denodo.com

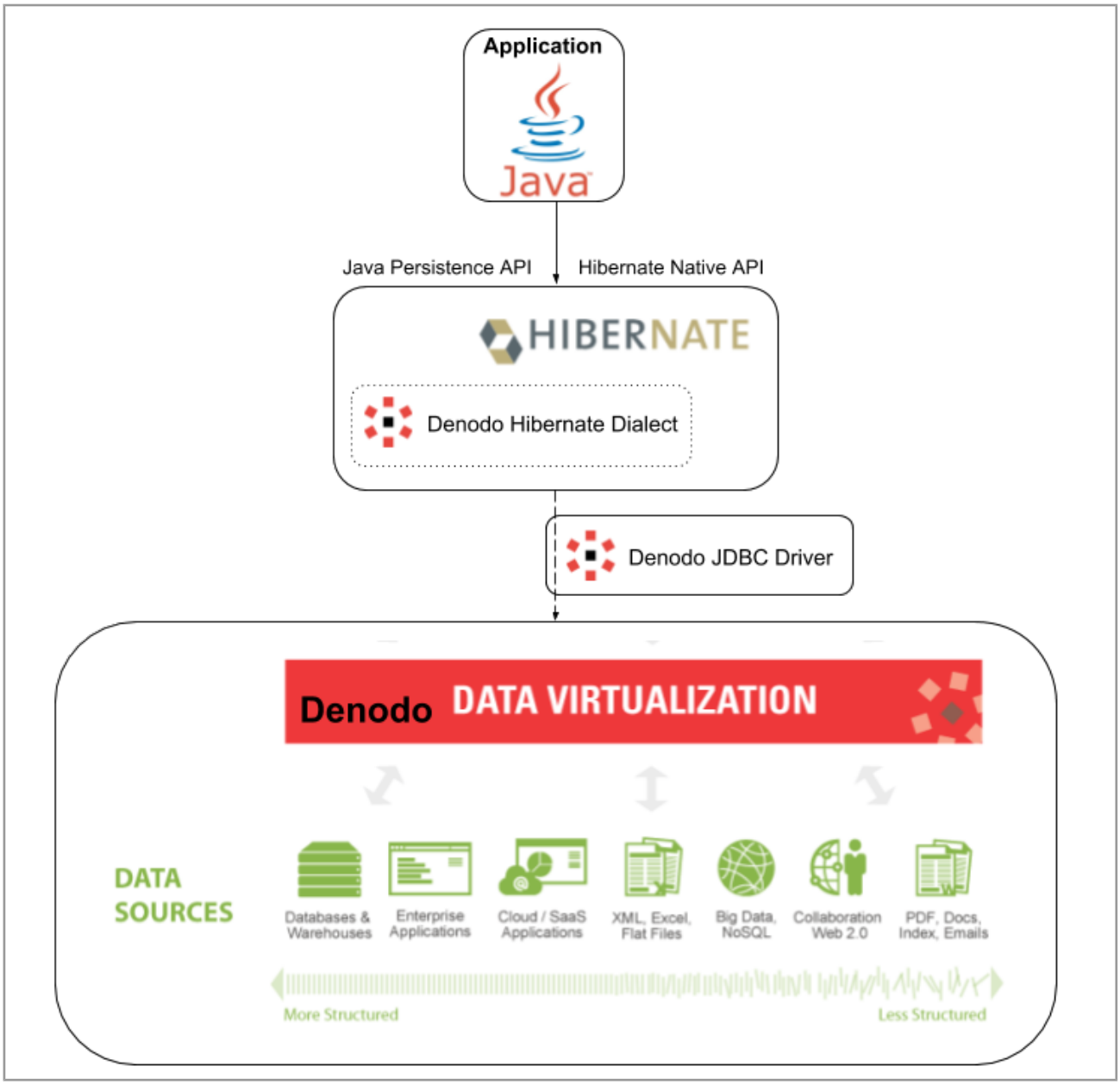
CONTENTS

1 OVERVIEW

Hibernate is a framework that provides object/relational mapping (ORM) services for relational databases.

Hibernate is database agnostic. This means that it can work with different databases. However, as databases implement slightly variations of SQL standard, Hibernate includes vendor specific **dialects** to handle these differences and accomplishes tasks like obtaining a primary key identifier or structuring a SELECT query.

The **Denodo Hibernate Dialect** is the dialect provided by Denodo to generate the appropriate VQL so Hibernate can deal with Denodo databases.



Hibernate - Denodo integration

Disclaimer: This dialect has some limitations due to the conceptual mismatch between an ORM and a data virtualization platform. Therefore, Denodo does not recommend the use of Hibernate, or any ORM, on top of Denodo.

2 CONFIGURATION

What follows is a guide with the configuration steps required to get the Denodo Hibernate Dialect running.

2.1 JAVA CONFIGURATION

2.1.1 Dependencies

You will need to have the following libraries in your classpath:

- Hibernate
- Denodo VDP JDBC driver
- Denodo Hibernate Dialect jar: `denodo-hibernate-dialect-<version>.jar` in the `/dist` directory of this distribution

2.1.2 Properties

You need to add the following property in your Hibernate configuration file:

dialect = com.denodo.connect.hibernate.dialect.DenodoDialect

besides the JDBC connection properties:

connection.driver_class = com.denodo.vdp.jdbc.Driver
connection.url = jdbc:vdb://localhost:9999/*database*
connection.username = ...
connection.password = ...

2.1.3 Entity Java class

If you want **the primary key value to be generated automatically** you will need to define your entities with Hibernate's Sequence generator**.** The only requirement for this strategy is that the name of the sequence follows the convention:

`denodo_data_source_name:sequence_name,`

being **`denodo_data_source_name`** the name given in Denodo to the data source that contains the entity tables and the sequence-support structures.

```
package ...;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
```

```
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;

@Entity
public class Person {

   @Id
   @GeneratedValue(strategy = GenerationType.SEQUENCE,
                   generator = "sequence-generator")
   @SequenceGenerator(name = "sequence-generator",
                   sequenceName = "denodo_dataSource_name:foo_sequence",
                   initialValue = 1, allocationSize = 1)
   private int id;
   private String name;
   private int age;
   .............
}
```

If you do **not** need the **primary key value to be generated automatically** you only have to use the `@Id` annotation that marks which entity field is the primary key and initialize its value in your application.

```
package ...;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import javax.persistence.SequenceGenerator;

@Entity
public class Person {

   @Id
   private int id;
   private String name;
   private int age;
   .............
}
```

2.1.3.1 Datetime mappings in Denodo 7

To persist the new datetime types introduced in Denodo 7 you have to use the following Java types in you entity Java classes:

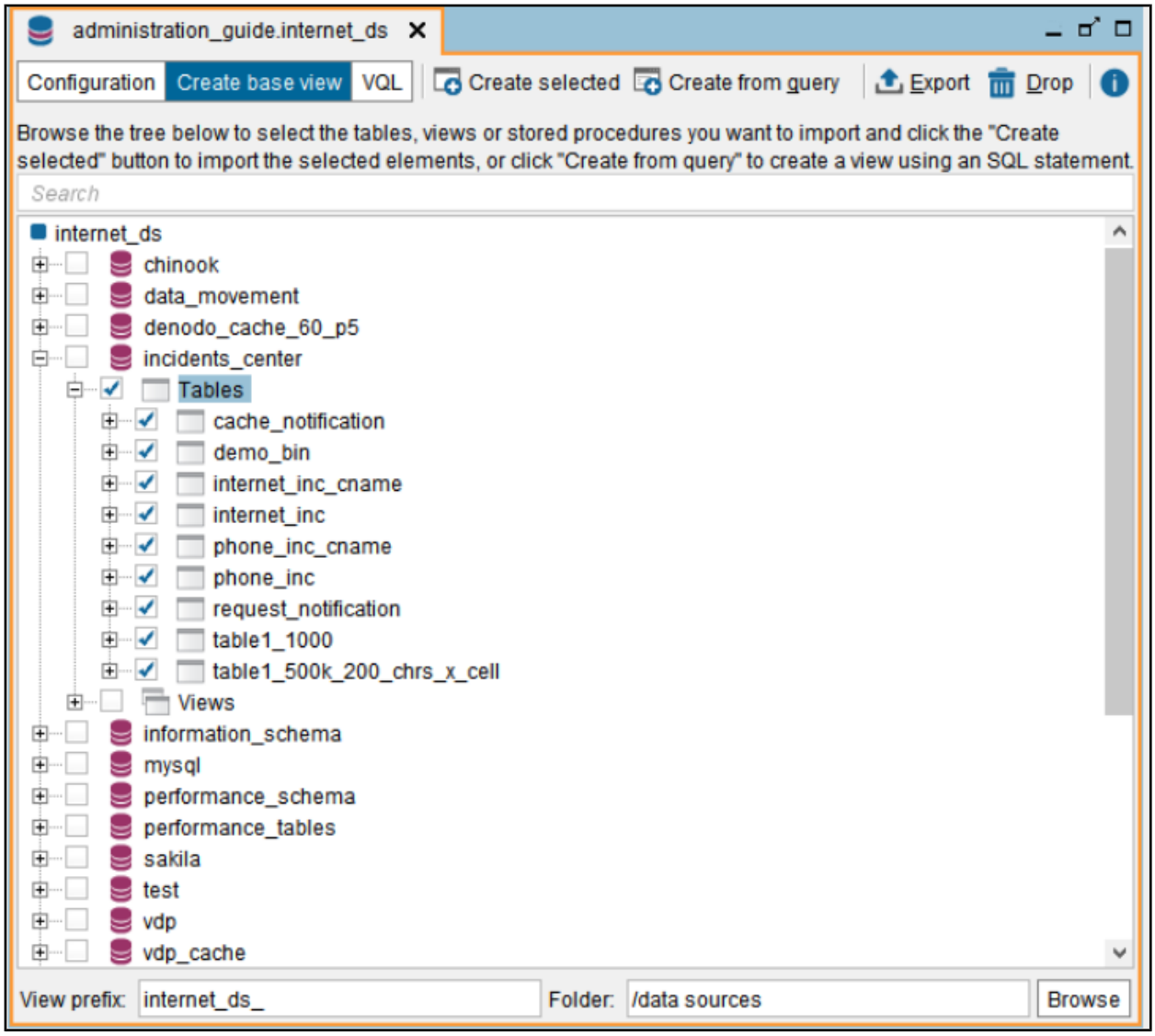
VDP Type	Java Type

localdate	java.time.LocalDate java.sql.Date
time	java.time.LocalTime java.sql.Time
timestamp	java.time.LocalDateTime
timestamptz	java.time.OffsetDateTime java.sql.Timestamp
intervalyearmonth	java.time.Period
intervaldaysecond	java.time.Duration

For more information on mappings between VDP datetime types and Java types, you can visit the Denodo 7 documentation: https://community.denodo.com/docs/html/browse/7.0/vdp/developer/access_through_jdbc/details_of_the_jdbc_interface/details_of_the_jdbc_interface#working-with-datetime-values-with-the-denodo-jdbc-driver

2.2 DENODO CONFIGURATION

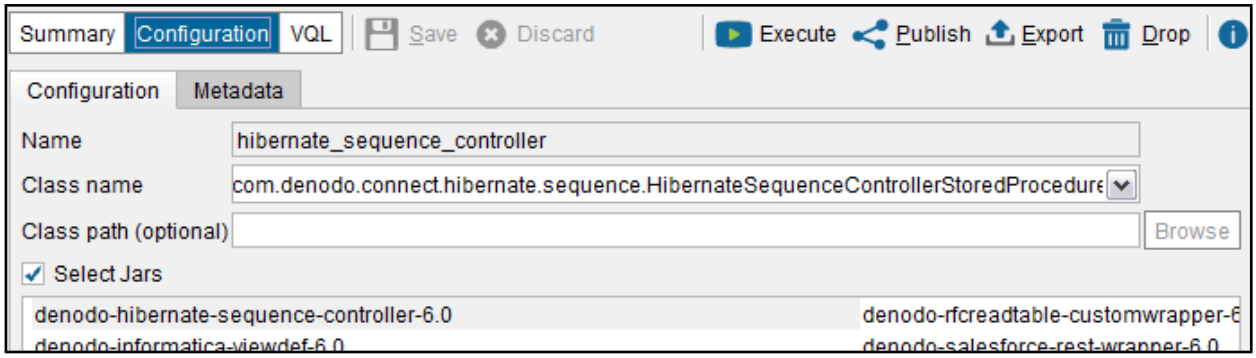
You need to **create the JDBC data source and the base views** that correspond with your Java entities, using the Virtual DataPort Administration Tool.



In order **to use the Sequence generator to generate primary key values**, you need to import the stored procedure `Hibernate Sequence Controller` in Denodo, (`denodo-hibernate-sequence-controller-<version>.jar` in the `/dist` directory of this distribution).

For this, in the Virtual DataPort Administration, click **Stored procedure** on the menu **File > New** and provide the following values:

- **Name**: `hibernate_sequence_controller`
- **Class name**: `com.denodo.connect.hibernate.sequence.HibernateSequenceControllerStoredProcedure`
- **Select Jars**: `denodo-hibernate-sequence-controller-<version>`



If you do **not** need the **primary key value to be generated automatically** you do **not** need to import the stored procedure `Hibernate Sequence Controller` in Denodo.

2.3 MYSQL CONFIGURATION

If you want the **primary key value to be generated automatically** and your data source is a MySQL you need to import the script `mysql_hibernate_sequence_definition.sql`,(in the `/scripts` directory of this distribution), into your MySQL server in order for the Sequence generator strategy to work properly.

The `mysql_hibernate_sequence_definition.sql` script defines several procedures and a table for generating unique values when invoked by the Denodo Hibernate Dialect.

2.4 ORACLE CONFIGURATION

If you want the **primary key value to be generated automatically** and your data source is an Oracle, the Denodo Hibernate Dialect will use the sequence object provided by Oracle.

You have to create the sequence manually, otherwise your Java application will throw the error:
`ORA-02289: sequence does not exist`

Being the configuration of the Sequence Generator, in the Entity Java class, something like this:

```
@SequenceGenerator(name = "sequence-generator",
                   sequenceName = "oracle_ds:foo_sequence",
                   initialValue = 1, allocationSize = 1)
```

you have two options for creating the sequence:

1. In the **Oracle** database

```
CREATE sequence "foo_sequence" START WITH 1 INCREMENT BY 1
```

2. In **Denodo** Administration Tool > VQL Shell > selecting the proper database

```
CALL hibernate_sequence_controller('create','oracle_ds:foo_sequence',1,1)
```

3 LIMITATIONS

- Note that `Denodo Hibernate Dialect` **does not support DDL operations**, as DDL in VQL is much more complex than in SQL.

  This means that establishing the property `hibernate.hbm2ddl.auto` to any of its possible values will have no effect.

- At the moment, **If you want the primary key value to be generated automatically** `Denodo Hibernate Dialect` only works with MySQL and Oracle data sources.

  As Denodo does not provide a method to generate unique integers on its own, the support of this dialect for generating primary keys depends highly on the underlying database and requires an intrusive setup on the data source. Losing therefore, one of the major advantages of Denodo that is abstracting where the data comes from.

- There are many ID generator strategies defined in Hibernate, but the Sequence is the only one supported by Denodo and therefore, by the `Denodo Hibernate Dialect`.

  And it only works for MySQL and Oracle, as stated in the previous point.