



## Active Record JDBC Adapter for Progress OpenEdge Databases

**Auteur** : [KANTENA](#) Paris XI France.  
**Date** : Septembre 2009  
**Licence** : MIT Licence  
**Contact** : [contact@kantena.com](mailto:contact@kantena.com)

*URL Plugin* :  
*URL Gem* :  
*HomePage* : [HomePage](#)

*Version* : 1.0 Beta

*Environnement* : Ruby on Rails , Jruby. Progress OpenEdge version 10 minimum

### Présentation

Adapteur jdbc permettant d'utiliser Active Record (AR) avec des bases Progress OpenEdge.

### Installation

Plugin : script/plugin install git://github.com/kantena/activerecord-openedge-adapter.git

### Spécificités

#### 0 – JDBC OpenEdge à inclure au projet

Il faut posséder les JAR du JDBC OpenEdge (Licence commercial)

**Il sont à placer dans le répertoire « lib » de votre projet rails.**

Les fichiers .jar suivants sont nécessaires :

- base.jar
- openedge.jar
- util.jar

#### 1 - Gestion des ids

Pour la gestion des ids, une table DUMMY\_SEQUENCE définie avec une seule colonne ainsi qu'une séquence DUMMY\_SEQUENCE doivent être créées.

La table doit contenir une et une seule ligne quelconque.

La séquence doit être spécifiée ainsi :

- incrément : 1
- start : 0
- nomaxvalue
- cycle : oui

**NOTE IMPORTANTE** : Cette table et cette séquence sont automatiquement créées par l'adapteur si elles n'existent pas, par les requêtes suivantes :

```
"create table pub.DUMMY_SEQUENCE (id integer)"
"insert into PUB.DUMMY_SEQUENCE(id) values(1)"
"create sequence pub.dummy_sequence increment by 1, start with 0, nomaxvalue, cycle"
```

## 2 - Gestion des dates

Pour assigner une date à un attribut AR qui correspond à une colonne de type date en Progress OpenEdge, il est conseillé d'utiliser la classe Date de Ruby plutôt que des chaînes de caractères ('01-10-2007').

### Exemple

Pour insérer le valeur date 20/09/2009

```
foo.date = Date.new(2009,9,20)
foo.save    # => 2009-09-20 dans la base
foo.reload
puts foo.date.strftime("%d/%m/%Y") # => 20/09/2009
```

Pour gérer le format d'affichage des dates voir Ruby On Rails Internationalization

## 3 - Gestion des champs "extent" de Progress OpenEdge

On accède et utilise , côté Ruby, les attributs qui proviennent d'une colonne extent Progress de la manière la plus naturelle qui soit :

### Exemple

*Progress* : libel (extent : 5, subtype integer)

**Utilisation en Ruby :**

### Affectation – Lecture

```
Affectation : libel[2] = 4
Lecture    : puts libel[2] # => 4
```

```
foo.libel[5] = 3
foo.save
foo.libel[5] = nil
foo.reload
puts foo[5] # => 3
```

### Sélection

```
result = find ( :all, :conditions => ["libel[2] = ? ",4])
```

**Attention** : libel[0] contient nil, et ne correspond pas à une sous colonne de libel côté Progress.

#### 4 - Database.yaml

**Option adapter du database.yaml => openedge**

Exemple :

```
test
adapter      : openedge
port         : 20001
username     : pca
password     :
host         : localhost
database     : approx
```

#### 5 - Find (:limit => , :offset =>)

La notion d'offset n'existe pas en Progress/SQL.

Ceci dit l'adaptateur permet tout de même *d'utiliser l'offset de la même manière qu'avec une autre base.*

Une méthode d'AR a été surchargée pour cela, les lignes du resultset correspondant à la valeur de l'offset ne sont pas instanciées en objets ruby et non retournées.

#### 6 - Migrations

##### 6.1 Colonnes Progress OpenEdge de type Extent

Lors de la migration d'une base Progress OpenEdge vers une autre base, la tâche rake de migration gère les colonnes progress de type "extent" de la manière suivante :

Exemple :

Progress : libel (extend : 5, subtype integer) , générera les colonnes :

libel\_\_1 (type : INTEGER)

...

libel\_\_5 (type : INTEGER)

##### 6.2 Colonnes CHAR x(n)

CHAR x(n) ==> VARCHAR(n/2 + 1)

## 7 – Type Casting

Progress OpenEdge	Progres/SQL	Rails
INTEGER	integer	:integer
CHAR	varchar	:string
INT64	bigint	:integer
DATE	date	:date
DECIMAL	numeric(,)	:decimal
LOGICAL	bit(1)	:boolean
DATETIME	timestamp	:datetime
DATETIME-TZ	timestamp_timezone	:timestamp
BLOB	lvarbinary(104857600)	:binary
CLOB	lvarchar(104857600)	:text
RAW	varbinary	:binary
RECID(déprécié)	integer(10)	:integer
ROWID	bigint	:integer

**Note :** Pas de type float en Progress.

## 7 - Problèmes connus

1) Le 'reload' d'un object AR ne fonctionne pas si le modèle utilise set\_primary\_keys pk1,pk2,.. (*DR Nic gem composite\_primary\_keys*)

Exemple :

foo.attr = 1; foo.save ; foo.reload => Error

La raison du bug a été localisée dans le gem composite\_primary\_keys.

2) Si vous migrez d' une autre base vers une base Progress OpenEdge, attention à ne pas avoir de noms de colonnes qui sont des mots réservés Progress (level, status,...) ou qui contiennent certains caractères spéciaux ('-' , ...)

3) FactoryGirl ne quote pas les noms de colonnes dans les conditions qu'elle passe au Find(...:conditions=> ,...), ce qui engendre une erreur SQL du JDBC OpenEdge si par exemple une des colonnes de la table se nomme 'level' ...

4) Les tâches rake ([db:migrate](#) , [db:test](#) ...) ne sont pas encore implémentées pour êtres utilisées avec des bases Progress OpenEdge

## 8 - En cours de développement

1) Tâches *rake* pour :

1. Migrer d'une base Progress vers MySql ou autre.
2. Lancer les tests unitaires, fonctionnels ...