INFOTEC DataBase Translation (iDBTrans)

Daniel A. Cervantes Cabrera

INFOTEC-CDMX
Julio 2020

Resumen

INFOTEC DataBase Translation (iDBTrans), es un programa desarrollado en la Dirección Adjunta de Innovación y Conocimiento del INFOTEC CDMX, el cual transforma la mayoría de los elementos de una base de datos creada con el sistema comercial Oracle [2] a sus equivalentes de PostgreSQL[3]. Representando así, una herramienta estratégica en la migración de aplicaciones en sectores públicos como privados que requieran liberarse del alto pago de licencia del uso de Oracle y usar una alternativa de código libre, el cual por sus características técnicas lo hacen uno los *softwares* más potentes del mercado. iDBTrans realiza transformaciones de sentencias y estructuras (incluyendo procedimientos almacenados, conocidos más comúnmente por su nombre en inglés *stored procedures*) SQL de Oracle a PostgreSQL. Para lo anterior, realiza un análisis léxico y construcción de árboles de parseo de las sentencias SQL por traducir, a continuación a través de recorridos recursivos remplaza y/o sustituye aquellos elementos no compatibles de una base a otra, basándose en diccionarios previamente definidos (los cuales se pueden enriquecer fácilmente). Esta metodología, permite obtener un sistema robusto y optimizado que compite en velocidad y precisión en la migración con los *softwares* comerciales similares como AWS de la empresa Amazon [4].

Un subproducto derivado de este desarrollo, es el llamado *INFOTEC-JDBC-POSTGRESQL*, el cual es un *software* tipo *driver*, enfocado al apoyo de la migración **no** de la base de datos, sino de los aplicativos. Permitiendo a los programadores muy poca o inclusive ninguna intervención en este proceso, esto se debe a que al remplazar el *driver* estándar Oracle JDBC [5] con el del INFOTEC-JDBC-POSTGRESQL cada ejecución (a través de `execute`, `executeQuery`, `executeUpdate`, etc.) de algún *query* o *update* de interacción con la base de datos Oracle en aplicativo, será traducido "al vuelo" a su respectivo de PostgreSQL, permitiendo ejecutar esta interacción a la base de PostgreSQL de manera transparente para el desarrollador.

Índice

1. Introducción

Debido a la gran cantidad de datos y transacciones que se manejan la mayoría de las empresas privadas o de la administración pública, es necesario contar con sistemas computacionales (aplicativos) basados en bases de datos (BD) que permitan la administración eficiente de consultas y actualizaciones de esta información. Uno de los gestores más comúnmente usados para este propósito es Oracle [2], desarrollado por la empresa *Oracle Corporation* el cual sigue el paradigma relacional y se considera como uno de los más completos, por su facilidad de uso, soporte de transacciones (*queries*), estabilidad y escalabilidad. Hace algunos años su dominio en el mercado era casi total, sin embargo en la actualidad debido a la gran competitividad con nuevos gestores, problemas en las nuevas versiones y su alto costo de licenciamiento, muchos usuarios están optando por nuevas alternativas. Dentro de éstas destaca PostgreSQL, programa de código abierto publicado bajo la licencia PostgreSQL [3] (similar a la BSD o MIT). El cual de acuerdo a sus autores, sus características técnicas

la hacen una de los gestores de bases de datos más potentes y robustos del mercado. Su desarrollo comenzó hace más de 16 años, y durante este tiempo, estabilidad, potencia, robustez, facilidad de administración e implementación de estándares han sido las características que más se han tenido en cuenta durante su desarrollo. PostgreSQL funciona muy bien con grandes cantidades de datos y una alta concurrencia de usuarios accediendo a la vez a el sistema.

Por lo anterior actualmente muchas instituciones o empresas requieren contar con herramientas que permitan realizar traducciones o migración de un gestor a otro. Existen diferentes tanto comerciales como código abierto, destacan por la parte comercial AWS DataBase Migration Service de Amazon [4] y por la parte de código libre oracle2postgres [6]. La primera tienen muchas funcionalidades que la hacen muy útil en el proceso de migración, sin embargo implica un costo de uso el cual depende de la cantidad de registros por migrar. En el caso del desarrollo de oracle2postgres (desarrollada en Python), se puede decir que se encuentra en una versión muy preliminar y actualmente no permite realizar una migración como se esperaría.

En el INFOTEC se desarrolló una aplicación de este tipo llamada iDBTrans, bajo una licencia de código libre LGPL [1], la cual a través de el análisis léxico de sentencias y estructuras SQL de Oracle por medio de recorridos recursivos, uso de diccionarios de tipos de datos y funciones, realiza la traducción de la mayoría de los elementos de la base de datos incluyendo los llamados procedimientos almacenados (*stored procedures*) a su respectivas sentencias SQL de PostgreSQL.

2. Análisis Léxico de sentencias SQL

Un analizador léxico y un *parser* son componentes de *software* que permiten procesar secuencias de caracteres. Así por ejemplo los compiladores e intérpretes incorporan estos elementos para la ejecución de los programas. Sin embargo existe una amplia gama de aplicaciones y en esta sección se describirá brevemente su uso para la migración de sentencias SQL.

Un analizador léxico permite "romper" o subdividir secuencias de caracteres en subsecuencias llamadas *tokens* y realiza una clasificación de éstos. Esta información es la entrada para el *parser*, el cual realiza un análisis por medio de una estructura de datos tipo árbol n-ário llamado árbol de parseo (*parser tree*). En general, el desarrollo de estos programas de forma eficiente puede llegar a ser complejo, por lo que es conveniente utilizar herramientas que permiten su generación automática, así se puede mencionar por ejemplo Yacc [9] y JavaCC [10], éste último diseñado para el lenguaje de programación JAVA el cual es la base de la biblioteca JSQLParser[11] extendida y utilizada para el desarrollo de iDBTrans.

Todo analizador léxico requiere la definición de una gramática que le permita identificar *tokens* y las estructuras de las cadenas de caracteres por procesar. Así a manera de ejemplo, supongamos que se desea procesar cadenas con ope-

raciones aritméticas básicas como:

$$99 + 42 + 15 \tag{1}$$

entonces utilizando la sintaxis de JavaCC, podríamos definir la siguiente gramática para este caso sencillo como:

```
SKIP : { " " }
SKIP : { "\n" | "\r" | "\r\n" }
TOKEN : { < PLUS : "+" > }
TOKEN : { < NUMBER : (["0"-"9"])+ > }
```

En donde la primer línea indica que el espacio en blanco es un *token*, pero no es considerado, es decir no será suministrado al *parser*. Similarmente la segunda línea indica lo mismo para los saltos de línea. La tercera indica que el símbolo de suma (+), será un *token* y establece un nombre simbólico como `PLUS`. Finalmente la cuarta línea representa una expresión regular para cualquier dígito y define la sintaxis usada para los números enteros dando el nombre simbólico `NUMBER` a este *token*. Indicando además (con el símbolo +) que podrá ser cualquier secuencia de uno o más dígitos. El token EOF (*end of file*) se genera automáticamente al finalizar la lectura del archivo analizado. Así al considerar la suma (1), el analizador léxico generaría la siguiente secuencia de *tokens*:

`NUMBER, PLUS, NUMBER, PLUS, NUMBER, EOF.`

Ahora el *parser* puede ser definido utilizando llamada notación Backus–Naur Form (BNF)[12], para gramáticas libres de contexto en la siguiente manera:

```
void Start():{}{<NUMBER> (<PLUS> <NUMBER>)* <EOF>}
```

Indicando que una secuencia es legítima siempre y cuando comienza con un *token* de `NUMBER`, termine con un `EOF` y entre éstos dos, cero o más secuencias de caracteres de `PLUS` seguido por un `NUMBER`. Un ejemplo del árbol de *parseo* para este caso, podría ser como se muestra a continuación:

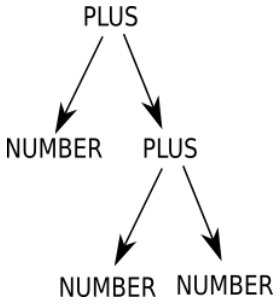


Figura 1: Árbol generado por *parser*.

Claramente el caso anterior, es muy simple si lo comparamos con el análisis léxico y construcción de *parser* para sentencias SQL, ya que esto implica la

definición de gramáticas muchos más amplias y compleja. Sin embargo una vez establecida (para este objetivo se utilizo la biblioteca JSQLParser la cual sin embargo tuvo que ser extendida ya que no consideraba muchos casos, principalmente el manejo de procedimientos almacenados) el procedimiento es análogo.

Así por ejemplo a continuación se muestra una sentencias SQL y su respectivo árbol de *parseo*:

```
SELECT id, name FROM t_user WHERE WHERE EF.ID_STATUS = 16
           AND CF.ID_STATUS = 16
           AND SYSDATE BETWEEN FEC_INICIAL AND FEC_FINAL
```

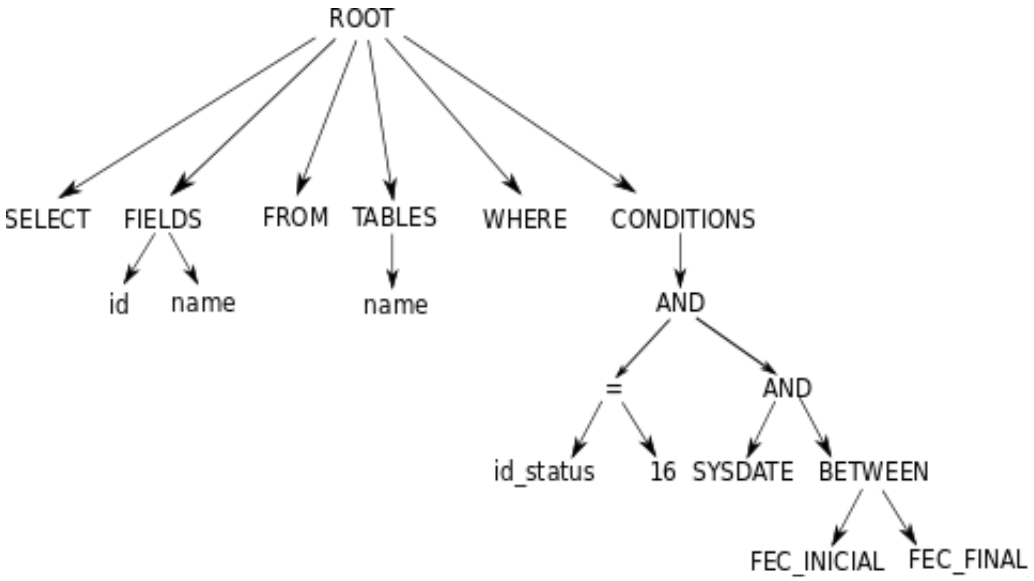


Figura 2: Árbol generado por *parser*.

Una vez construidas estos árboles de *parseo*, el proceso de migración requiere el recorrido recursivo de éstos por medio de los llamados *visitors*, también generados automáticamente por JavaCC, a través de los cuales es posible realizar los remplazos y/o sustituciones de sintaxis no compatibles para cada gestor. Por ejemplo en la sentencia SQL anterior, es necesario realizar un remplazo de la porción: `SYSDATE BETWEEN FEC_INICIAL AND...`, debido a que ni la función `SYSDATE` ni `BETWEEN` son reconocidos por postgreSQL, así para este caso, fue necesario realizar un implementación particular `INFOTEC_POSTGRES.SYSDATE` y remplazar `BETWEEN` como se muestra a continuación:

Oracle:

```
WHERE EF.ID_STATUS = 16 AND CF.ID_STATUS = 16 AND SYSDATE BETWEEN
FEC_INICIAL AND FEC_FINAL
```

PostgreSQL:

```
Postgres:
WHERE EF.ID_STATUS = 16 AND CF.ID_STATUS = 16 AND
INFOTEC_POSTGRES.SYSDATE() >= FEC_INICIAL
AND INFOTEC_POSTGRES.SYSDATE() <= FEC_FINAL;
```

3. Análisis y diseño en iDBTrans

En esta sección se describen brevemente los módulos orientados a objetos principales (ver diagrama UML figura 3), que componen el sistema iDBTrans. Destacando los módulos de representación de la base de datos genérica `GeneralDB`, a partir de la cual es posible derivar cualquier tipo de base dependiendo del gestor que se utilice, notemos que tiene como atributo principal un conector el cual gestiona la conexión por medio del *driver* proporcionado, también se muestran los métodos públicos para procesamiento de *queries* y *updates*. El módulo `DBDictionary`, representa el diccionario a partir del cual se derivan los casos particulares para cada gestor, éstos son "alimentados" por diccionarios tipo JSON los cuales pueden ser fácilmente extendidos. Finalmente `DBMigrator` que como su nombre lo indica, se encarga del proceso de migración, por lo que tiene como atributos principales los diccionarios y las base de datos de cada gestor por migrar. Además, contiene métodos los cuales una vez establecidas la conexión carga la información de los diccionarios y la base de datos fuente (oracle), comienzan a realizar la migración de tablas, vistas, procedimientos almacenados etc., como se describió en la sección anterior. Toda esta migración es almacenada en otra estructura de datos que representa la versión migrada de la base original, a partir de la cual se generan los *scripts* de salida.

4. Instalación y Uso

4.1. Instalación

Uno de los principales objetivo en el desarrollo de la aplicación iDBTrans, es que su uso sea simple y que no requiera necesariamente un especialista en el área de base de datos para poder realizar la migración.

Así este proceso requiere de los siguientes elementos:

1. Tener instalado a menos la versión 8 de Java en el equipo de cómputo que realizará la migración.
2. Base de datos de Oracle que por migrar (fuente).
3. Base de datos de salida de PostgreSQL que contendrá versión migrada (salida).

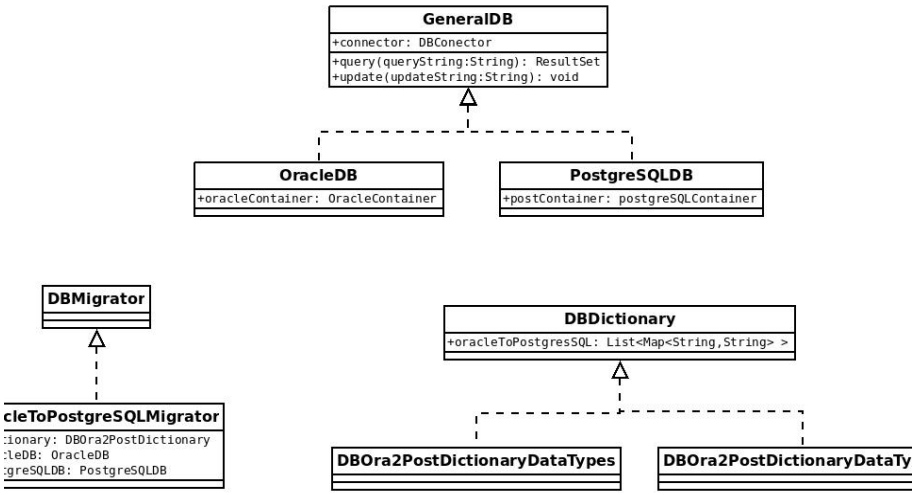


Figura 3: Diseño UML para iDBTrans.

El punto 1 se describe en el anexo 1 y la instalación y creación de la base de datos de salida de PostgreSQL en el anexo 2 de este documento. Para el punto 2, se está suponiendo que el usuario ya tiene instalada y cargada la base de datos de Oracle ya sea en su máquina local y/o en un servidor con los permisos necesarios para su acceso.

4.2. Uso de la versión de línea de comandos

Una vez que se cuenta con los puntos descritos en la sección anterior, es posible realizar la migración. Así para poder procesar la información de las bases de datos de los puntos 2 y 3 se requiere la creación de dos archivos de configuración que llamaremos `config_source.json` y `config_target.json` de la base de datos fuente y salida respectivamente, cuya estructura deberán ser como se muestra a continuación:

```
{
    "url":"jdbc:oracle:thin:@HOST_NAME:PORT:SDI",
    "user":"YOUR_USERNAME",
    "password":"YOUR_PASSWORD",
    "schema":"YOUR_DATABASE_SCHEMA_NAME_TO_MIGRATE"
}
```

Listing 1: config_source.json

```
 {
    "url":"jdbc:postgresql://HOST_NAME:PORT/DATABASENAME",
    "user":"YOUR_USERNAME",
    "password":"YOUR_PASSWORD",
    "driver":"org.postgresql.Driver",
    "schema":"YOUR_DATABASE_SCHEMA_NAME_MIGRATED"
```

```
7 }
```

Listing 2: config_target.json

Una vez que estos archivos están definidos la ejecución del programa se puede realizar desde la línea de comandos escribiendo:

```
1 java -jar idbtrans.jar
```

Una salida típica después del comando anterior se muestra a continuación:

```
-------------------------------------------------------------------------------------
--   ______   ____    ____    ______
--  /\__  _\ /\  _`\ /\  _`\ /\__  _\
--  \/_/\ \/ \ \ \/\ \ \ \L\ \/_/\ \/ _ __    __      ___     ____
--     \ \ \  \ \ \ \ \ \  _ <' \ \ \/\`'__\/'__`\  /' _ `\  /',__\
--      \_\ \__\ \ \_\ \ \ \L\ \ \ \ \ \//\ \L\.\_/\ \/\ \/\__, `\
--      /\_____\\ \____/\ \____/  \ \_\ \_\\ \__/.\_\ \_\ \_\/\____/
--      \/_____/ \/___/  \/___/    \/_/\/_/ \/__/\/_/\/_/\/_/\/___/
--
--                           Version  0.1.0
--                        DAIC -  INFOTEC CDMX

-------------------------------------------------------------------------------------

Loading dictionary...
Loading source data base file configuration...
Loading tarjet data base file configuration...
Connecting to source data base...
Connecting to tarjet data base...
Started at:Fri Mar 27 03:36:43 CST 2020 getting sequences...
Started at:Fri Mar 27 03:36:43 CST 2020 getting default values column...
Started at:Fri Mar 27 03:36:44 CST 2020 getting partitions...
Started at:Fri Mar 27 03:36:45 CST 2020 getting tables...
Started at:Fri Mar 27 03:36:45 CST 2020 Getting columns...
Started at:Fri Mar 27 03:37:55 CST 2020 getting Views...
Started at:Fri Mar 27 03:37:55 CST 2020 getting Materialized Views...
Started at:Fri Mar 27 03:37:55 CST 2020 getting Primary Keys...
Started at:Fri Mar 27 03:37:57 CST 2020 getting Foreign keys...
Started at:Fri Mar 27 03:39:32 CST 2020 getting constraints type check...
Started at:Fri Mar 27 03:39:33 CST 2020 getting indexes...
Started at:Fri Mar 27 03:39:33 CST 2020 getting objects code, procedures,packages,funtions...
Started at:Fri Mar 27 03:39:36 CST 2020 getting dbaObjects...
Started at:Fri Mar 27 03:39:36 CST 2020 getting pck procedures....
Started at:Fri Mar 27 03:39:36 CST 2020 getting code procedures package...
Started at:Fri Mar 27 03:39:38 CST 2020 Set code to procedures and functions ..
Started at:Fri Mar 27 03:39:52 CST 2020 Set code to triggers..
Started at:Fri Mar 27 03:39:53 CST 2020 Set objects type user defined for every schema..
Started at:Fri Mar 27 03:39:53 CST 2020 Setting status VALID or INVALID for objects of code
Start of migration...
Creating schemas and script for PostgreSQL..
Creating Sequences and script for PostgreSQL..
Creating tables and script for PostgreSQL..
Creating default values for tables and script for PostgreSQL..
Creating partitions and script for PostgreSQL..
Creating Indexes and script for PostgreSQL..
Creatting Functions and script for PostgreSQL..
Creatting Procedures and script for PostgreSQL..
Creatting packages and script for PostgreSQL..
Creatting triggers and script for PostgreSQL..
Creatting views and script for PostgreSQL..
Creatting materialized views and script for PostgreSQL..
Inserting data on tables for PostgreSQL..
Started at:Fri Mar 27 03:40:14 CST 2020
SCHEMA: SIO inserted rows:4
No. of Schemas: 1
----------------------------------------
```

```
Infotec_Ora2Post: Migration Report
----------------------------------------
        Input Data Base
----------------------------------------
********** BEGIN SCHEMA: SIO [1] de [1] **********
No. of Tables to migrate : 807
No. of Indexes to migrate : 354
No. of Column Default: 603
No. of Views: 141
Status: VALIDS 122, INVALIDS 19
No. of Materilized Views: 39
Status: VALIDS 0, INVALIDS 39
No. of Functions: 996
Status: VALIDS 900, INVALIDS 96
No. of Procedures: 509
Status: VALIDS 374, INVALIDS 135
Packages :35 VALIDS : 24 INVALIDS:11 No. of Function and Procedures VALIDS: 366
No. of Triggers: 122 VALIDS: 80 INVALIDS:42
********** END SCHEMA SIO **********
----------------------------------------
        Output Data Base
----------------------------------------
No. of Schemas: 1
********** BEGIN SCHEMA: SIO [1] de [1] **********
No. of Tables: 824
        Status: Migrated 806 No migrated 18
No. of Index: 354
        Status: Migrated 311, No migrated 43
No. of Column Default: 603
        Status: Migrated 595, No migrated 8
No. of Views: 141
        Status: Migrated 89, No migrated 52 Converted:122 Not-converted:19
No. of Materialized Views: 39
        Status: Migrated 19, No migrated 20 Converted:30 Not-converted:9
No. of Functions: 996
        Status: Migrated 834, No migrated 162 Converted:0 Not-converted:996
No. of Procedures: 509
        Status: Migrated 298, No migrated 211 Converted:368 Not-converted:141
No. of Functions and Procedures Packages: 35
ALL PROCEDURES MIGRATED :0 NO MIGRATED:0
No. of Triggers: 122
        Status: Migrated 70, No migrated 52 Converted:80 Not-converted:42
********** END SCHEMA SIO **********
----------------------------------------
Ended process...
Thanks...
```

Aquí se puede observar el proceso completo de migración, desde la carga de los diferentes diccionarios, elementos de básicos de las tablas, vistas, procedimientos almacenados etc., de cada esquema, así como la migración de cada uno de éstos. Finalmente se presentan las estadísticas de la base de datos de salida.

4.2.1. Exportación de datos

Una vez finalizada la migración, es posible exportar los de datos, es decir, realizar un copia de la información contenida en las todas (o algunas) tablas de la base de datos de origen a la de destino. Como parte de este proyecto, se desarrolló una aplicación enfocada para este propósito. Para su ejecución es necesario definir una archivo de configuración (se puede generar desde la interfaz gráfica) en donde se establece la estructura de la base de datos, junto con

las tablas y el número de registros por exportar (parámetro `rowsToExport`), a continuación se muestra un ejemplo:

```
{
  "lstSchemasTable" : [ {
    "schemaName" : "YOUR_SCHEMA_NAME",
    "tableList" : [ {
      "tableName" : "TABLE_NAME",
      "columnList" : [ {
        "schemaName" : "YOUR_SCHEMA_NAME",
        "tableName" : "TABLE_NAME",
        "columnName" : "COLUMN_NAME",
        "dataType" : "DATA_TYPE",
        ...
           }, {
        "schemaName" : "YOUR_SCHEMA_NAME",
        "tableName" : "TABLE_NAME",
        "columnName" : "COLUMN_NAME",
        "dataType" : "DATA_TYPE",
        ...
      }]
      "fkList" : [ ],
      "constraintList" :"" ,
      "comment" : "",
      "status" : "VALID",
      "statusText" : "",
      "hintStatus" : "",
      "sqlQuery" : "",
      "temporary" : false,
      "owner" : "YOUR_SCHEMA_NAME",
      "numRows" : "Number",
      "partitioned" : "NO",
      "rowsToExport" : "Number",
      "exportSelected" : true

      ...
    }]
    ...
  }]
...
}
```

Listing 3: config_target.json

Una vez creado el archivo anterior, es posible ejecutar el programa de exportación desde la línea de comandos escribiendo:

```
java -jar idbtrans-rwr.jar
```

A continuación se deberá imprimir una salida similar a la siguiente:

```
--------------------------------------------------------------------------------------------------------
--     ____    ____   ______
--  __/\  _`\ /\  _`\ /\__  _\
-- /\_\ \ \/\ \ \ \L\ \/_/\ \/ _ __     __      ___     ____          _ __   __  __  __  _ __
-- \/\ \ \ \ \ \ \  _ <' \ \ \/\`'__\/'__`\  /' _ `\  /',__\  _______/\`'__\/\ \/\ \/\ \/\`'__\
--  \ \ \ \ \_\ \ \ \L\ \ \ \ \ \ \//\ \L\.\_/\ \/\ \/\__, `\/\______\ \ \/ \ \ \_/ \_/ \ \ \/
--   \ \_\ \____/\ \____/  \ \_\ \_\\ \__/.\_\ \_\ \_\/\____/\/______/\ \_\  \ \___x___/'\ \_\
--    \/_/\/___/  \/___/    \/_/\/_/ \/__/\/_/\/_/\/_/\/___/           \/_/   \/__//__/   \/_/
--
--                           Version  0.0.0
--                        DAIC -  INFOTEC CDMX
--
--------------------------------------------------------------------------------------------------------
Loading source data base file configuration...
Loading tarjet data base file configuration...
Connecting to source data base...
Connecting to tarjet data base...
Select and inserting data on tables from Oracle to PostgreSQL..
Started at:Thu Jul 16 23:24:17 CDT 2020
Inserting in Table CC_CLASE_AUXILIAR with 13 rows.
Inserting in Table ASIENTOS_DOCUMENTO with 256 rows.
Inserting in Table AD_USUARIO_ENTIDAD with 178 rows.
Inserting in Table CG_ARTICULO with 16 rows.
Inserting in Table CG_DEDUCCION_AUMENTO with 55 rows.
Inserting in Table CG_CONTRIBUYENTE with 1 rows.
Inserting in Table CG_CONCEPTO with 736 rows.
Inserting in Table CG_CATALOGO with 14 rows.
Inserting in Table CG_BENEFICIARIO with 11567 rows.
Inserting in Table CG_BANCOS with 27 rows.
Inserting in Table CC_PLAN_CUENTA with 695 rows.
Inserting in Table CC_NORMA_CONTABLE with 1 rows.
Inserting in Table CC_MATRIZ_RETENCIONES_CONTA with 46 rows.
Inserting in Table CC_MATRIZ_RELACION with 55 rows.
Inserting in Table CG_SUBCONCEPTO with 755 rows.
Inserting in Table CG_SERVICIO_ENTIDAD with 20 rows.
Inserting in Table CG_SERVICIO with 20 rows.
Inserting in Table CG_SECUENCIA_PROCESO with 7 rows.
Inserting in Table CG_EJERCICIO with 6 rows.
Inserting in Table CG_DOCUMENTO_RESPALDO with 215 rows.
Inserting in Table CG_DETALLE_SERVICIO_ENTIDAD with 143 rows.
Inserting in Table CG_DETALLE_SERVICIO with 151 rows.
Inserting in Table CG_DETALLE_CATALOGO with 1105 rows.
Inserting in Table CG_DEDUCCION_AUMENTO_ENTIDAD with 51 rows.
Inserting in Table TE_LBANCOS with 0 rows.
Inserting in Table TE_FORMATO_CHEQUE with 0 rows.
Inserting in Table TE_DETALLE_PAGO with 10 rows.
Inserting in Table TE_CUENTAS_TESORERIA with 6274 rows.
Inserting in Table DETALLE_DOCUMENTO with 3043 rows.
Inserting in Table DETALLE_CONCEPTO with 3 rows.
Inserting in Table DEDUCCIONES_DOCUMENTO with 4 rows.
Inserting in Table CT_OPERACIONES_BANCARIAS with 6 rows.
Inserting in Table CP_TIPO_GASTO with 3 rows.
Inserting in Table CP_RECURSO_AUXILIAR with 181 rows.
Inserting in Table CP_RECURSO with 61 rows.
Inserting in Table CP_PROYECTO with 1 rows.
Inserting in Table CP_OBJETO_GASTO with 613 rows.
Inserting in Table CP_ESTRUCTURA_CLAVE_EGRESOS with 36 rows.
Inserting in Table CG_TIPO_EXPEDIENTE with 10 rows.
Inserting in Table CG_TIPO_DOCUMENTO_PROCESO with 323 rows.
Inserting in Table CG_TIPO_DOCUMENTO_EVENTO with 70 rows.
Inserting in Table CG_TIPO_DOCUMENTO with 22 rows.
Inserting in Table CG_TIPO_DOC_IDENTIFICACION with 2 rows.
Inserting in Table CG_SUBFUNCION with 111 rows.
Inserting in Table CG_EJERCICIO_ENTIDAD with 12 rows.
Inserting in Table SG_OPERACIONES with 770747 rows
Ended process...
Thanks...
```

4.3. Uso de la versión *Web*

Adicionalmente a la versión de línea de comandos, se desarrolló una aplicación *Web* que tiene el propósito de facilitar el uso de iDBTrans. En este caso la información de conexión de las bases origen y de salida se especifica a través de los campos de texto en la vistas mostradas en la figura 4.

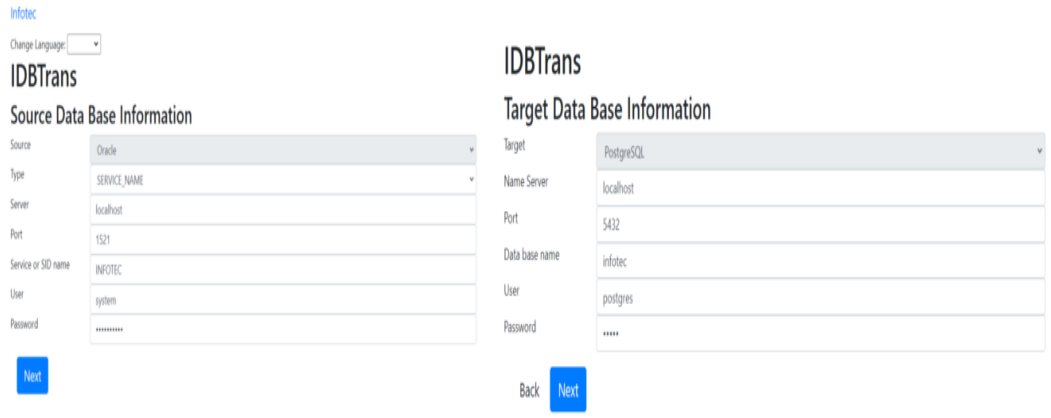


Figura 4: Vistas para conexión de base de datos.

Una vez establecida la comunicación se procede a seleccionar el (o los) esquema(s) por migrar y se realiza un *clic* en el botón **Next** como se muestra en la figura 5.

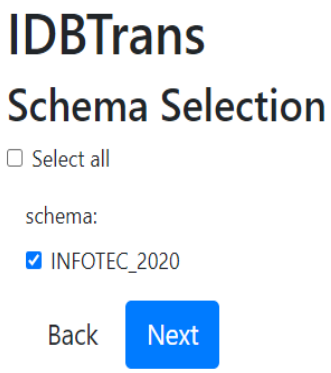
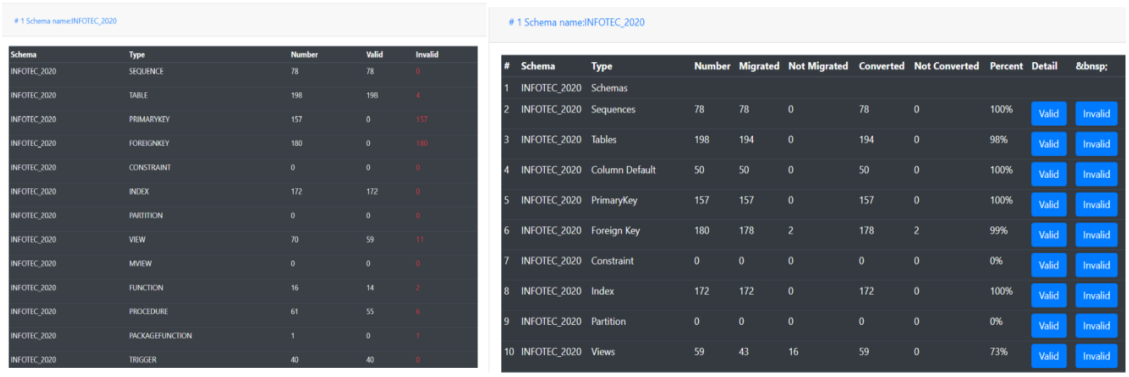


Figura 5: Selección de esquema y botón de inicio de migración (**Next**).

A continuación se mostrará la información de la estructura de la base de datos por migrar y sus estadísticas de migración (ver figura 6).

4.3.1. Exportación de datos

La exportación de los datos para la versión *Web* permite seleccionar las tablas y el número de registros por migrar (vista superior izquierda de la figura

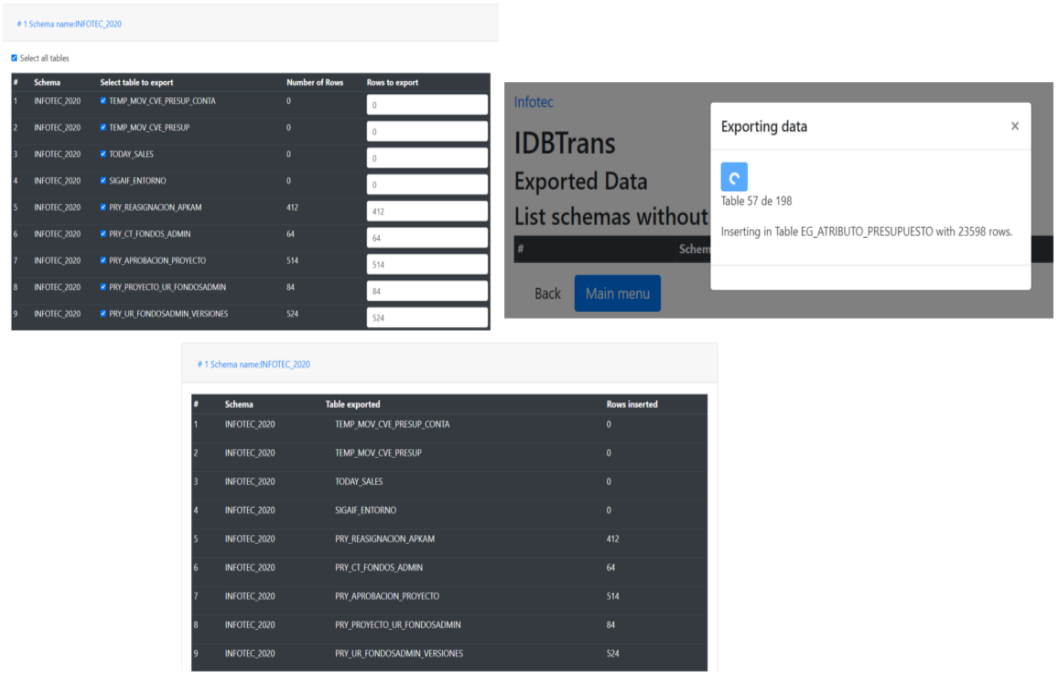


Schema	Type	Number	Valid	Invalid
INFOTEC_2020	SEQUENCE	78	78	0
INFOTEC_2020	TABLE	198	198	4
INFOTEC_2020	PRIMARYKEY	157	0	157
INFOTEC_2020	FOREIGNKEY	180	0	180
INFOTEC_2020	CONSTRAINT	0	0	0
INFOTEC_2020	INDEX	172	172	0
INFOTEC_2020	PARTITION	0	0	0
INFOTEC_2020	VIEW	70	59	11
INFOTEC_2020	MVIEW	0	0	0
INFOTEC_2020	FUNCTION	16	14	2
INFOTEC_2020	PROCEDURE	61	55	6
INFOTEC_2020	PACKAGEFUNCTION	1	0	1
INFOTEC_2020	TRIGGER	40	40	0

#	Schema	Type	Number	Migrated	Not Migrated	Converted	Not Converted	Percent	Detail	&bnsp;
1	INFOTEC_2020	Schemas								
2	INFOTEC_2020	Sequences	78	78	0	78	0	100%	Valid	Invalid
3	INFOTEC_2020	Tables	198	194	0	194	0	98%	Valid	Invalid
4	INFOTEC_2020	Column Default	50	50	0	50	0	100%	Valid	Invalid
5	INFOTEC_2020	PrimaryKey	157	157	0	157	0	100%	Valid	Invalid
6	INFOTEC_2020	Foreign Key	180	178	2	178	2	99%	Valid	Invalid
7	INFOTEC_2020	Constraint	0	0	0	0	0	0%	Valid	Invalid
8	INFOTEC_2020	Index	172	172	0	172	0	100%	Valid	Invalid
9	INFOTEC_2020	Partition	0	0	0	0	0	0%	Valid	Invalid
10	INFOTEC_2020	Views	59	43	16	59	0	73%	Valid	Invalid

Figura 6: Información de datos por migrar y estadísticas de migración.

4), una vez definidos, se realiza el proceso de exportación de la información (vista superior derecha) y finalmente se despliegan un reporte de la de este proceso (vista inferior).



#	Schema	Table exported	Rows inserted
1	INFOTEC_2020	TEMP_MOV_CVE_PRESUP_CONTA	0
2	INFOTEC_2020	TEMP_MOV_CVE_PRESUP	0
3	INFOTEC_2020	TODAY_SALES	0
4	INFOTEC_2020	SIGAIF_ENTORNO	0
5	INFOTEC_2020	PRY_REASIGNACION_APKAM	412
6	INFOTEC_2020	PRY_CT_FONDOS_ADMIN	64
7	INFOTEC_2020	PRY_APROBACION_PROYECTO	514
8	INFOTEC_2020	PRY_PROYECTO_UR_FONDOSADMIN	84
9	INFOTEC_2020	PRY_UR_FONDOSADMIN_VERSIONES	524

Figura 7: Vista (superior izquierda) para selección de registros por migrar, Vista (superior derecha) registro de proceso de copia y vista (inferior) de despliegue de información copiada.

5. Resultados

En esta sección se describen las estadísticas de la bases de datos migradas, para sistemas de INFOTEC, SEMARNAT y CONDUCEF. En donde se puede observar un alto porcentaje de migración la cual es competitiva (y en algunos casos mejor) que el *software* comercial de Amazon [4].

```
ESTADICAS POR BASE DE DATOS

BASE INFOTEC

-- Generated by IDBTRANS:Thu Jul 23 11:37:29 CDT 2020 Version: 1.0.1
-- Copyright 2020 INFOTEC - DAIC - CDMX. All rights reserved.

********** INPUT ORACLE BEGIN SCHEMA[1] of [1]: INFOTEC_2020 **********
No. of Sequences to migrate : 78
No. of Tables to migrate : 194 Temporary:4
No. of Column Default: 50
No. of Primay Keys: 157
No. of Foreign Keys: 180
No. of Constraints: 0
No. of Indexes to migrate : 172
No. of Partition: 0
No. of Views: 70
Status: VALIDS 59, INVALIDS 11
No. of Materilized Views: 0
Status: VALIDS 0, INVALIDS 0
No. of Functions: 16
Status: VALIDS 14, INVALIDS 2
No. of Procedures: 61
Status: VALIDS 55, INVALIDS 6
Packages :1 VALIDS : 0 INVALIDS:1 No. of Function and Procedures VALIDS: 0
No. of Triggers: 40 VALIDS: 40 INVALIDS:0
********** END SCHEMA INFOTEC_2020 **********

----------------------------------------

********** OUTPUT POSTGRESQL BEGIN SCHEMA [1] of [1]: INFOTEC_2020  **********
No. of Sequences: 78
Status: Migrated 78 No migrated 0
No. of Tables: 194
Status: Migrated 194 No migrated 0
No. of Column Default: 50
Status: Migrated 50, No migrated 0
No. of Primay Keys: 157
Status: Migrated 157, No migrated 0
No. of Foreign Keys: 180
Status: Migrated 178, No migrated 2
No. of Constraints: 0
Status: Migrated 0, No migrated 0
No. of Index: 172
Status: Migrated 172, No migrated 0
No. of Partition: 0
Status: Migrated 0, No migrated 0
No. of Views: 59
Status: Migrated 43, No migrated 16 Converted:59 Not-converted:0
No. of Materialized Views: 0
Status: Migrated 0, No migrated 0 Converted:0 Not-converted:0
No. of Functions: 14
Status: Migrated 11, No migrated 3 Converted:14 Not-converted:0
No. of Procedures: 55
Status: Migrated 52, No migrated 3 Converted:55 Not-converted:0
No. of Packages: 0
No of Packages:0 ALL Procedures:0 Migrated :0 No-Migrated:0 Converted:0  Not-Converted:0
No. of Triggers: 40
Status: Migrated 40, No migrated 0 Converted:40 Not-converted:0
```

```
********** END SCHEMA INFOTEC_2020 **********

BASE CONDUSEF

- Generated by IDBTRANS:Thu Jul 23 11:42:29 CDT 2020 Version: 1.0.1
-- Copyright 2020 INFOTEC - DAIC - CDMX. All rights reserved.

********** INPUT ORACLE BEGIN SCHEMA[1] of [1]: SIO **********
No. of Sequences to migrate : 33
No. of Tables to migrate : 807 Temporary:17
No. of Column Default: 603
No. of Primay Keys: 207
No. of Foreign Keys: 331
No. of Constraints: 0
No. of Indexes to migrate : 515
No. of Partition: 4
No. of Views: 141
Status: VALIDS 122, INVALIDS 19
No. of Materilized Views: 39
Status: VALIDS 30, INVALIDS 9
No. of Functions: 997
Status: VALIDS 901, INVALIDS 96
No. of Procedures: 509
Status: VALIDS 374, INVALIDS 135
Packages :35 VALIDS : 24 INVALIDS:11 No. of Function and Procedures VALIDS: 366
No. of Triggers: 122 VALIDS: 80 INVALIDS:42
********** END SCHEMA SIO **********

----------------------------------------

********** OUTPUT POSTGRESQL BEGIN SCHEMA [1] of [1]: SIO  **********
No. of Sequences: 33
Status: Migrated 33 No migrated 0
No. of Tables: 807
Status: Migrated 807 No migrated 0
No. of Column Default: 603
Status: Migrated 595, No migrated 8
No. of Primay Keys: 207
Status: Migrated 206, No migrated 1
No. of Foreign Keys: 331
Status: Migrated 327, No migrated 4
No. of Constraints: 0
Status: Migrated 0, No migrated 0
No. of Index: 515
Status: Migrated 512, No migrated 3
No. of Partition: 4
Status: Migrated 0, No migrated 4
No. of Views: 122
Status: Migrated 95, No migrated 27 Converted:122 Not-converted:0
No. of Materialized Views: 30
Status: Migrated 12, No migrated 18 Converted:30 Not-converted:0
No. of Functions: 901
Status: Migrated 837, No migrated 64 Converted:901 Not-converted:0
No. of Procedures: 374
Status: Migrated 311, No migrated 63 Converted:374 Not-converted:0
No. of Packages: 24
No of Packages:24 ALL Procedures:366 Migrated :227 No-Migrated:139 Converted:365  Not-Converted:1
No. of Triggers: 80
Status: Migrated 72, No migrated 8 Converted:80 Not-converted:0
********** END SCHEMA SIO **********
BASES SEMARNAT
BASE SISSAO (Multiples esquemas)

-- Generated by IDBTRANS:Thu Jul 23 11:50:44 CDT 2020 Version: 1.0.1
-- Copyright 2020 INFOTEC - DAIC - CDMX. All rights reserved.

********** INPUT ORACLE BEGIN SCHEMA[1] of [11]: UCD_USDB **********
```

```
No. of Sequences to migrate : 28
No. of Tables to migrate : 59 Temporary:0
No. of Column Default: 54
No. of Primay Keys: 16
No. of Foreign Keys: 1
No. of Constraints: 0
No. of Indexes to migrate : 26
No. of Partition: 0
No. of Views: 7
Status: VALIDS 7, INVALIDS 0
No. of Materilized Views: 0
Status: VALIDS 0, INVALIDS 0
No. of Functions: 3
Status: VALIDS 1, INVALIDS 2
No. of Procedures: 34
Status: VALIDS 34, INVALIDS 0
Packages :9 VALIDS : 1 INVALIDS:8 No. of Function and Procedures VALIDS: 2
No. of Triggers: 22 VALIDS: 22 INVALIDS:0
********** END SCHEMA UCD_USDB **********
----------------------------------------

********** OUTPUT POSTGRESQL BEGIN SCHEMA [1] of [11]: UCD_USDB  **********
No. of Sequences: 28
Status: Migrated 28 No migrated 0
No. of Tables: 59
Status: Migrated 59 No migrated 0
No. of Column Default: 54
Status: Migrated 53, No migrated 1
No. of Primay Keys: 16
Status: Migrated 16, No migrated 0
No. of Foreign Keys: 1
Status: Migrated 1, No migrated 0
No. of Constraints: 0
Status: Migrated 0, No migrated 0
No. of Index: 26
Status: Migrated 24, No migrated 2
No. of Partition: 0
Status: Migrated 0, No migrated 0
No. of Views: 7
Status: Migrated 2, No migrated 5 Converted:7 Not-converted:0
No. of Materialized Views: 0
Status: Migrated 0, No migrated 0 Converted:0 Not-converted:0
No. of Functions: 1
Status: Migrated 0, No migrated 1 Converted:1 Not-converted:0
No. of Procedures: 34
Status: Migrated 33, No migrated 1 Converted:34 Not-converted:0
No. of Packages: 1
No. of Function and Procedures in Package TOOLKIT [1]: 2
Status: Migrated 2, No migrated 0 Converted:2 Not-converted:0
No of Packages:1 ALL Procedures:2 Migrated :2 No-Migrated:0 Converted:2  Not-Converted:0
No. of Triggers: 22
Status: Migrated 14, No migrated 8 Converted:22 Not-converted:0
********** END SCHEMA UCD_USDB **********
```

6. El driver Infotec PostgreSQL

Un subproducto del desarrollo de iDBTrans, fue la extensión del driver `jdbc_postgres` de postgreSQL [13] llamado `infotec_jdbc_postgres`, al cual se le incorporaron módulos de análisis léxico y parseo, de tal manera que todas las invocaciones de sentencias son traducidas inmediatamente. Esto permite facilitar la migración de aplicativos desarrollados en JAVA que utilicen el driver `jodbc` Oracle, ya que al remplazarlo por `infotec_jdbc_postgres`, todas las consultas serán traducidas automáticamente de manera transparente para el desarrollador, permitiendo así, establecer las comunicaciones previas de Oracle, pero ahora a su respectiva versión migrada de PostgreSQL.

En la siguiente porción de código, se ejemplifica este proceso:

```
1   //Se invoca al driver infotec_jdbc_postgres
2   Class.forName("org.infotec_postgresql.Driver");
3
4   //Se establece la conexi'on a la base de datos de
       PostgreSQL migrada
5   c = DriverManager.getConnection("jdbc:infotec_postgresql:
6   //localhost:5432/dvdrental","postgres", "postgres");
7
8   System.out.println("Successfully Connected.");
9
10  stmt = c.createStatement();
11
12  String sentence = " SELECT film.film_id,
       film.title,inventory_id FROM film, inventory WHERE
       inventory.film_id = film.film_id(+);";
13
14
15  ResultSet rs = stmt.executeQuery( sentence );    //Query
       Oracle traducido autom'aticamente a postgreSQL
16  while ( rs.next() ) {
17    String  title = rs.getString("title");
18    int inventory_id  = rs.getInt("inventory_id");
19    System.out.printf( "title = %s , Inventory_id = %s ",
         title,inventory_id );
20  }
21
22  rs.close();
23  stmt.close();
24  c.close();
```

7. Modelo de Negocio

Debido a que el desarrollo de iDBTrans tiene una licencia LGLP es de acceso libre bajo los lineamientos de ésta. El modelo de negocio propuesto está basado en los siguientes puntos:

- Consultoría del uso especializado de iDBTrans.

- Exportación optimizada de registros a gran escala.
- Acompañamiento completo en el proceso de migración.
- Migración completa de aplicaciones que utilicen el gestor Oracle al de PostgreSQL.

8. Participantes

Este desarrollo ha contado con la partición del Dr. Feliú D. Sagols Troncoso y el Lic. Marcos Romero Sandoval de la Dirección Adjunta de Innovación y Conocimiento del INFOTEC-CDMX.

9. Informe de avances y evidencia

En este trabajo se ha desarrollado una herramienta que permite la migración de la mayoría de los elementos de la estructura de una base diseñada en Oracle a PostgreSQL. Este proyecto se encuentra en su etapa final, ya que están implementadas prácticamente la mayoría de sus funcionalidades tanto de la versión en línea de comandos como la interfaz de usuario (versión Web), sin embargo es necesario continuar con pruebas exhaustivas de integración. Es decir, realizar una mayor cantidad de migraciones de base de datos, por lo que sería importante establecer colaboración con la Dirección Adjunta de Desarrollo de Software (DADS) para asesorarlos en el uso del *iDBTrans* y obtener retroalimentación de ellos, lo mismo para dependencias externas. Todos los códigos se encuentra alojados en el repositorio de la DADS (figura 8):

`http://gitlab.dads.infotec.mx/DAIC_IDB/DAIC_IDB`

10. Trabajo Futuro

Como trabajo futuro se plantea el desarrollo de un *Open DataBase Connectivity* (ODBC) que incorpore el proceso de migración de manera similar al caso del driver descrito en la sección anterior, sin embargo al hacerlo al nivel de esta capa, el lenguaje de programación con el que se desarrolló la aplicación es indistinto, permitiendo así abordar una mucho mayor gama aplicativos que requieran migración de sus sistemas de base de datos.

Para lo anterior se planea utilizar el proyecto UnixODBC [8] -una implementación ODBC de código abierto- y continuar con la metodología empleada para el desarrollo de iDBTrans, es decir, incorporar módulos de análisis léxico, parseo y diccionarios de tal manera que a través de los recorridos sobre el árbol de parseo para las sentencias SQL, se realicen los remplazos correspondientes de PostgreSQL. Haciendo transparente para el usuario la conexión y comunicación con la base de datos (ver figura 8).

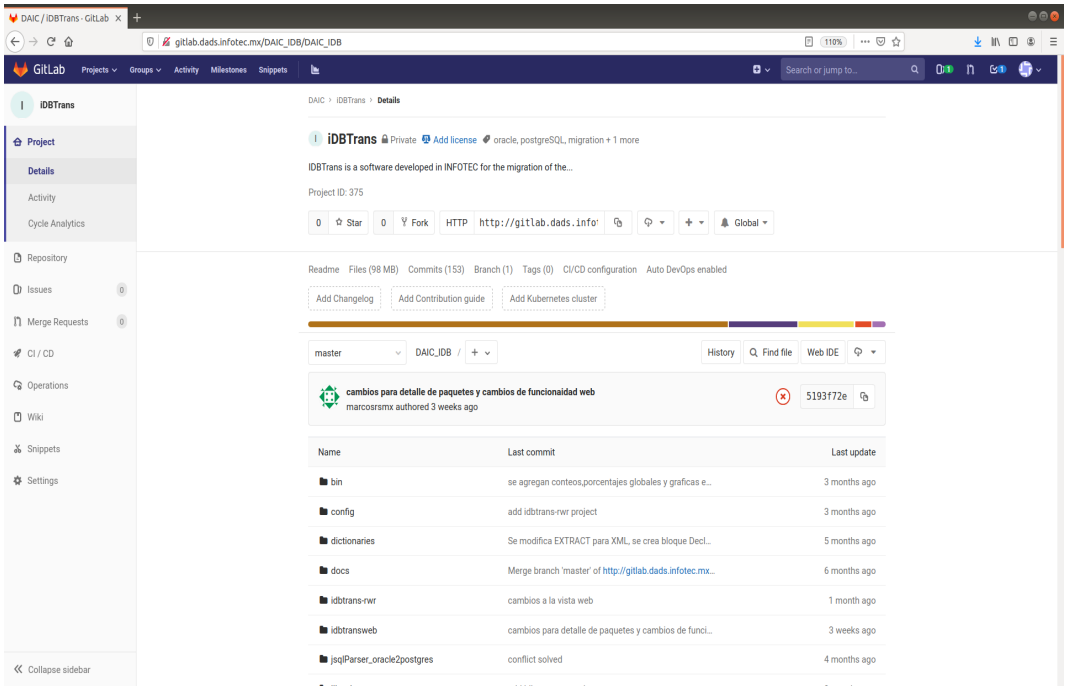


Figura 8: Repositorio gitlab de la DADS de iDBTrans.

11. Bibliografía

Referencias

[1] https://www.gnu.org/licenses/lgpl-3.0.html

[2] https://www.oracle.com/

[3] https://www.postgresql.org/

[4] https://aws.amazon.com/es/dms/

[5] https://docs.oracle.com/en/database/oracle/oracle-database/20/jjdbc/toc.htm

[6] https://pypi.org/project/oracle2postgres/

[7] https://about.gitlab.com/blog/2014/09/29/gitlab-flow/

[8] http://www.unixodbc.org/

[9] https://www.epaperpress.com/lexandyacc/

[10] https://javacc.github.io/javacc/

[11] https://github.com/JSQLParser/JSqlParser

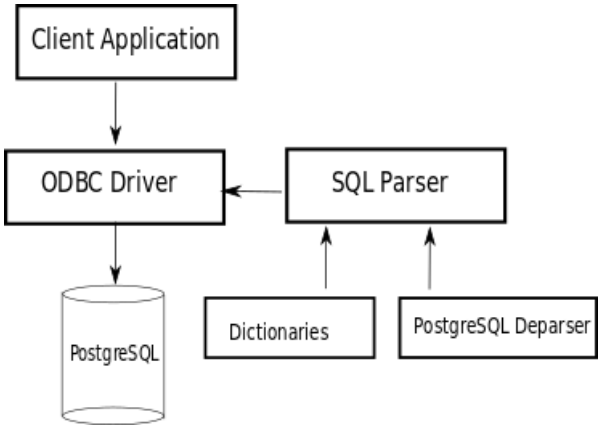


Figura 9: iDBTrans ODBC.

[12] http://www.garshol.priv.no/download/text/bnf.html

[13] https://jdbc.postgresql.org/download.html