SINCRONIZACIÓN DE DATOS ENTRE SISTEMAS DE PRODUCCIÓN Y ENSAYO

José Jesús Sánchez Farías

Tecnológico Nacional de México en Celaya

jesus.sanchez@itcelaya.edu.mx

Susana Guadalupe Rojas Serrano

Tecnológico Nacional de México en Celaya

susirojas60@gmail.com

Juan Ignacio Cerca Vázquez

Tecnológico Nacional de México en Celaya

nacho@itcelaya.edu.mx

Luis Alberto López González

Tecnológico Nacional de México en Celaya

luislao@itcelaya.edu.mx

Rubén Torres Frías

Tecnológico Nacional de México en Celaya

ruben.torres@itcelaya.edu.mx

Resumen

En equipos de desarrollo de software profesionales, existes diferentes ambientes en los cuales se pueden probar cambios de código fuente, realizar pruebas de funcionalidad o cambios experimentales, para esto, existen principalmente tres entornos: Desarrollo, Ensayo y Producción. Esta investigación se centra en cómo establecer una comunicación y sincronización de información entre los entornos de producción y ensayo, esto debido a la problemática que enfrenta un equipo de desarrollo de software para realizar pruebas de nuevas

funcionalidades con información real y confiable, y no con datos ficticios. Realizar estas pruebas con datos reales utilizados en un sistema en producción sobre un escenario de ensayo, permite a los usuarios y clientes del sistema probar las nuevas funcionalidades en un ambiente lo más real posible y así detectar posibles errores que se puedan presentar cuando éstas se trasladen a producción en un entorno real. Existen muchas técnicas que permiten la comunicación y sincronización de información entre sistemas desde servicios web, restauraciones de base de datos, hasta comunicación directa entre base de datos. La investigación se centra en la comunicación de dos bases de datos por dblink, un paquete que pertenece al popular gestor de base de datos libre llamado PostgreSQL, utilizado actualmente por empresas, instituciones públicas y privadas para sus sistemas en entornos de producción, además de estudiantes durante el aprendizaje.

Palabra(s) Clave: Dblink, Ensayo, Postgresql, Producción, Sincronización.

Abstract

In professional software development teams, there are different environments in which you can test source code changes, perform functionality tests or experimental changes, for this, there are mainly three environments: Development, Staging and Production. This research focuses on how to establish communication and synchronization of information between production and staging environments, due to the problems faced by a software development team to test new functionalities with real and reliable information, and not with fictitious data. Carrying out these tests with real data used in a production system on a staging scenario allows the users and clients of the system to test the new functionalities in an environment as real as possible and to detect possible errors that may occur when they are transferred to production in a real environment. There are many techniques that allow the communication and synchronization of information between systems as web services, database restorations and direct communication between databases. The research focuses on the communication of two databases by "dblink", a package belonging to the popular free database

manager PostgreSQL, currently used by companies, public and private institutions for their production systems, as well as students in learning environments.
Keywords: *Dblink, Postgresql, Production system, Stage system, Synchronization.*

1. Introducción

En empresas pequeñas y medianas especializadas en desarrollo de software es muy común que manejen tres ambientes que por los cuales pasa un software antes de llegar a las manos del usuario final, estos son:

- Desarrollo. En el ambiente de desarrollo (development) se maneja una copia del código del proyecto, los cambios que hacen los desarrolladores día a día son desplegados en éste para que nuevas características sean probadas y se vayan integrando en el producto a generar. Este ambiente se actualiza rápidamente y contiene la versión más reciente de una aplicación, normalmente el usuario final no tiene acceso a este.
- Ensayo. En el ambiente de ensayo (*staging*) se pone en marcha y despliega la aplicación previa a su liberación final, corresponde la versión candidata a liberar. Contiene la siguiente versión de una aplicación en la cual se harán las pruebas de estrés final. Es común que en este ambiente accedan usuarios finales y administradores para su aprobación antes de liberarse en el entorno de producción.
- Producción. En el ambiente de producción (*production*) se despliega la versión actual y más reciente de una aplicación. Es la aplicación que utilizan los usuarios finales en un entorno real donde se llevan a cabo las actividades del día a día del proceso automatizado. En esta versión preferentemente no se permiten cambios del código fuente excepto en fecha programada para liberación de una nueva versión:
    - ✓ Debian. Es una distribución del popular sistema operativo GNU/Linux, por lo tanto, se considera un sistema operativo completo que contiene software y sistemas basados en el kernel de Linux o FreeBSD. Permite la instalación de software libre normalmente proveniente del proyecto GNU. Debian es gratuito y de libre

descarga e instalación, se usa tanto en ambientes de desarrollo, ensayo y producción; incluso llega a usarse por programadores y administradores como sistema operativo de uso cotidiano en computadoras personales.

- ✓ Apt. Es un administrador de paquetes que permite la gestión del software que se instala en el sistema operativo Debian. Los paquetes pueden descargarse desde los discos de instalación o de servidores espejo distribuidos en diferentes países por todo el mundo. Al instalar un paquete de software por apt, este normalmente administra todas las dependencias del software a instalar, ahorrando este trabajo al usuario. Los comandos utilizados para la gestión de los paquetes son: apt-get para distribuciones Debian 8 o menores y simplemente apt para distribuciones Debian 9.
- ✓ PostgreSQL. Es un sistema gestor de base de datos objeto-relacional basado en POSTGRES. Este último fue desarrollado en el departamento de ciencias computacionales de la Universidad de California en Berkeley PostgreSQL es de código abierto y soporta la mayor parte del estándar SQL ofreciendo características importantes y modernas como: consultas complejas, llaves foráneas, triggers, vistas, transacciones y control de concurrencia. PostgreSQL puede ser utilizado, modificado y distribuido por cualquier persona libre de cargos para cualquier propósito como puede ser en industrias privadas, entornos comerciales o académicos. Al igual que Debian este motor de base de datos es utilizado en ambientes de desarrollo, ensayo y producción.
- ✓ DBLink. Es un módulo que soporta conexiones a otras bases de datos PostgreSQL dentro de una sesión de base de datos en un servidor. Las bases de datos con quien se conecta pueden ser locales o incluso remotas hospedadas en otros servidores con diferente ubicación geográfica. Para realizar la conexión se utiliza un usuario y contraseña previamente creados en el motor PostgreSQL a

conectar, así como los permisos necesarios sobre los objetos a acceder desde el servidor origen. Se puede acceder a tablas, vistas, funciones y en general cualquier objeto definido previamente. El uso más común de DBLink es la compartición de información entre sistemas de información, se ofrece a través de un medio fácil, rápido, eficiente y seguro para la transferencia de los datos.

- ✓ PL/PGSQL. Es un lenguaje procedural para el gestor de base de datos PostgreSQL. Puede ser utilizado para crear funciones y procedimientos desencadenadores, agrega estructuras de control al lenguaje SQL y se pueden ejecutar procedimientos que impliquen cálculos complejos. Con pl/pgsql se pueden agrupar bloques de instrucciones y consultas SQL dentro del servidor de base de datos para su posterior ejecución desde otros procedimientos, clientes de base de datos o aplicaciones externas al gestor. En PostgreSQL 9.0 o superior pl/pgsql está instalado por default, aunque aún se puede utilizar cómo un módulo cargable.

2. Metodología

La comunicación y sincronización de información entre dos bases de datos que se encuentran en entornos diferentes como producción y ensayo implica la instalación y configuración de una serie de paquetes y herramientas que permitirán dicha función. Para cumplir el objetivo, también se necesitan el hardware adecuado que permita la simulación de ambos entornos, para esto se utilizaron dos computadoras como servidores con características y arquitecturas adecuadas para las pruebas. En la figura 1 se muestra un diagrama que representa el esquema de trabajo de los servidores utilizados para realizar las pruebas de sincronización de información.

Los pasos que se realizaron para llevar a cabo la comunicación y sincronización entre las bases de datos, así como puesta en marcha de los servidores:

- Instalación y configuración del motor de bases de datos. En esta etapa se describe el procedimiento que permite la instalación y puesta en marcha del

motor de base de datos PostgreSQL dentro de un sistema operativo Linux Debian 9. El método utilizado para este proceso es a través de paquetes binarios instalados a través de la herramienta apt. Los siguientes comandos se ejecutan tanto en el servidor de producción como de ensayo:

- ✓ Instalación de motor de bases de datos y herramientas cliente:

  apt install postgresql postgresql-client

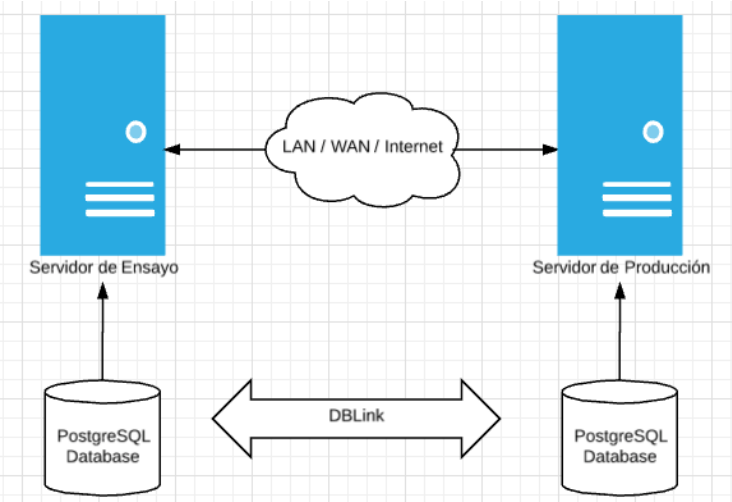

Figura 1 Diagrama servidores de ensayo y producción.

Una vez instalado PostgreSQL se realiza una prueba de funcionamiento con los siguientes comandos:

- ❖ Verificación de puerto. Mediante el comando telnet se verifica si el puerto 5432 de PostgreSQL está escuchando:

  $ telnet localhost 5432

- ❖ Verifica Proceso en ejecución. Mediante el comando ps se verifica que el proceso del servicio de PostgreSQL está ejecución y funcionado correctamente:

  $ ps -aux | grep postgres

- ✓ Conexión y listado de bases de datos. Mediante el comando psql se realiza una conexión al servicio y se muestra el listado de bases de datos existentes por default:

  # su - postgres

$ psql -l

- ✓ Creación de usuario. Se crea un usuario administrador para realizar la sincronización de información:
  $ createuser -s -d -P syncadmin
- ✓ Habilitar conexiones remotas:
  # joe/etc/postgresql/9.6/main/postgresql.conf
- ✓ Retirar los comentarios de las siguientes líneas y modificar:
  listen_ad = `*`
- ✓ Habilitar conexiones remotas para el usuario syncadmin y base de datos Adventureworks:
  # joe /etc/postgresql/9.6/main/pg_hba.conf
- ✓ Agregar la siguiente línea:
  host Adventureworks syncadmin 172.20.11.0/24 md5

- Montaje de base de datos en entornos de producción y ensayo. Para realizar pruebas reales de sincronización de información, es necesario tener las bases de datos correspondientes con los diferentes objetos que la representan. Para esto, se hará uso de una base de datos robusta con miles de registros para simular el entorno lo más real posible.

  La base de datos a utilizar tiene como nombre "Adventureworks", la cual es pública y disponible pasa su descarga [Adventureworks Database, 2017]. Esta base de datos maneja información de diferentes áreas de una empresa como pueden ser: recursos humanos, producción, personas, compras y ventas de productos. Para el ejemplo demostrativo y documentación de esta investigación nos enfocamos en el área de clientes y órdenes de compra de los clientes. El diagrama correspondiente se muestra en la figura 2.

  Dado que la base de datos en el servidor de producción simula un ambiente real de un sistema de información, está contendrá la base de datos completa, es decir, se monta tanto el esquema como la información misma. Para el caso del servidor de ensayo, contiene una base de datos con el

mismo esquema de base de datos, pero sin información, esto para realizar la sincronización de información entre ambos.

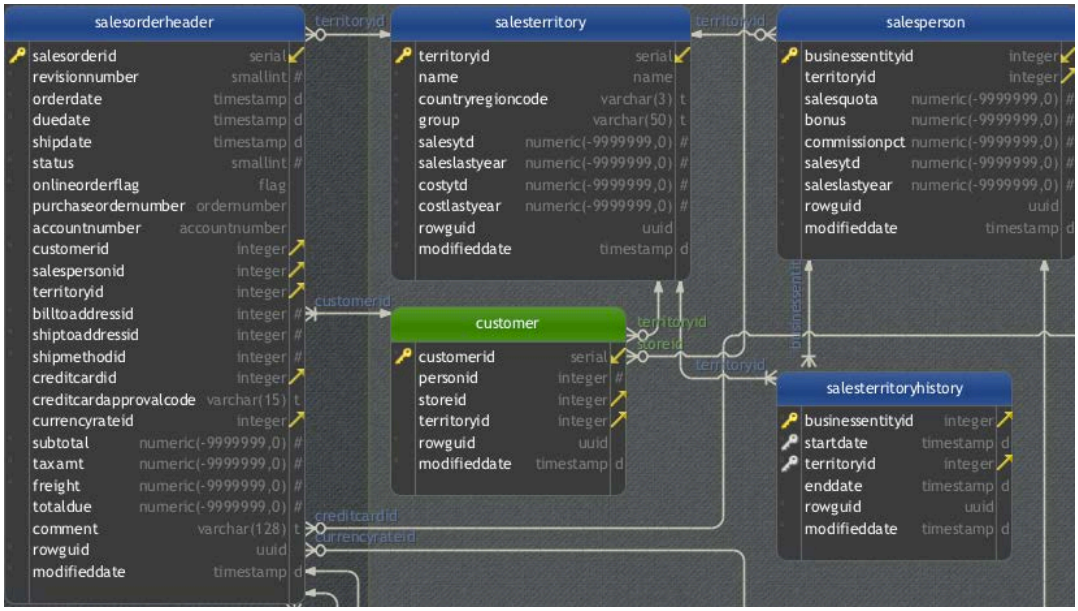

Figura 2 Diagrama relacional de base de datos.

El montaje del esquema de base de datos e información se realiza de la siguiente manera:

- ✓ Descargar base de datos y scripts de Adventureworks, colocar en un directorio en particular, ejemplo:

  /home/syncadmin/Downloads.

- ✓ Descomprimir con:

  unzip Adventureworks.zip

- ✓ Modificar archivos CSV para trabajarlos con PostgreSQL:

  ruby update_csvs.rb Crear base de datos e importar información:

  psql -c "CREATE DATABASE "Adventureworks";"

  psql -d Adventureworks < install.sql

Para el caso del servidor de ensayo contiene exactamente el mismo esquema e información excepto los catálogos a sincronizar.

- Instalación del paquete de comunicación. Una vez instalado el motor de base datos, se prosigue a la instalación y configuración del paquete de

comunicación dblink, dicho paquete no está habilitado por default, por lo tanto, se tendrá que seguir un procedimiento preciso para ponerlo en marcha. Este paso se puede realizar en un solo servidor y depende de la dirección de sincronización, para el ejercicio se instala en el servidor de ensayo ya que este será quien realice la sincronización de información con el servidor de producción.

Dentro de la carpeta */usr/share/postgresql/9.6/extension* se encuentra el script que permite agregar la extensión dblink a la base de datos Adventureworks de PostgreSQL, para esto, se ejecutan los siguientes comandos:

$ psql Adventureworks < /usr/share/postgresql/9.6/extension/dblink--1.2.sql
&a psql Adventureworks
Adventureworks=> create extension dblink;
Adventureworks=> dx

En la figura 3 se muestra la salida una vez habilitada la extensión dblink.

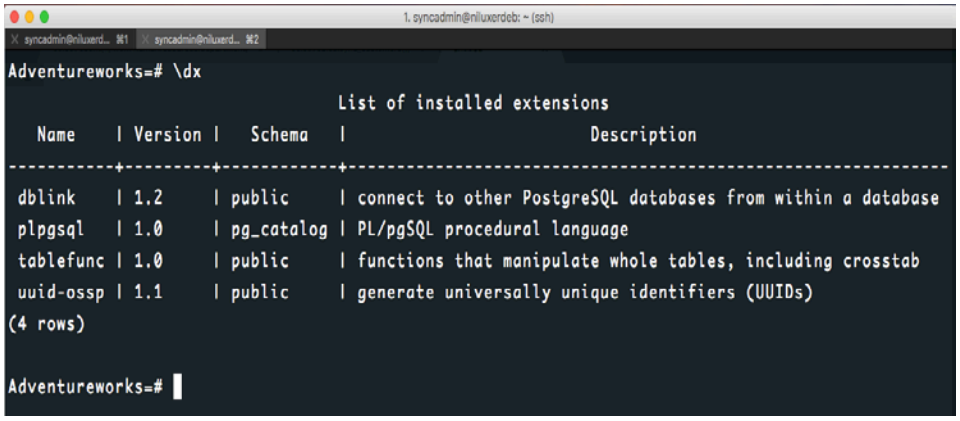

Figura 3 Listado de extensiones instaladas en PostgreSQL.

- Conexiones para la comunicación entre servidores. Antes de proseguir con la sincronización de información, se deben establecer las comunicaciones entre los servidores que hospedan las bases de datos a sincronizar. Se establecerá un esquema seguro y confiable para lograr una comunicación estable y precisa. Ambos servidores deben ser visibles por la red a la cual

se hayan conectado, puede ser una red LAN, WAN o una comunicación por Internet.

Una vez creadas las bases de datos y agregada la extensión para dblink, se procede a establecer conexiones que prueben la comunicación entre servidores y bases de datos. Dentro de un cliente de base de datos conectado al servidor de ensayo, se genera la consulta de la figura 4, la cual permite probar la conexión de bases de datos.

```
1 SELECT cust.customerid, cust.personid, cust.storeid
2 FROM
3    dblink(
4     'dbname=Adventureworks host=192.168.1.70 user=syncadmin password=nopasswd',
5     'select customerid, personid, storeid from sales.customer') as cust(customerid integer,
      personid integer, storeid integer)
```

Figura 4 Consulta para la conexión entre bases de datos.

- Técnica de sincronización de información. Se definen una serie de scripts en un lenguaje procedural pl/pgsql propio de PostgreSQL donde se codifican las diferentes consultas que particularmente sincronizan la información entre tablas dentro de las bases de datos. Con esto logrando la transferencia de información entre ambos entornos de producción y ensayo.

  Esta sección se divide principalmente en dos partes. La primera de ellas corresponde a creación de un script que permita la sincronización de la tabla *customer*, esto es, se enviarán los datos de los clientes ubicados en la base de datos de producción hacia la tabla de clientes ubicada en la base de datos de ensayo. En la segunda parte se realizan las mismas acciones, pero ahora enfocado a la sincronización de la tabla que corresponde a las órdenes de compra de los clientes, tabla *salesorderheader*.

  El procedimiento general para la sincronización de la tabla "customer" consiste en lo siguiente:

  - ✓ Crear una función de nombre sync_customer dentro del esquema sales.

- ✓ Realizar la conexión a la base de datos Adventureworks ubicada en el servidor de producción.
- ✓ Seleccionar todos los registros de la tabla sales.customer ubicada en el servidor de producción.
- ✓ Se sincronizan las siguientes columnas: customerid, personid, storeid, territoryid y modificationdate.
- ✓ Recorrer los registros uno por uno e insertarlos en la tabla del mismo nombre, pero ahora en el servidor de ensayo.
- ✓ Llevar un contador de registros para proporcionarlo como valor de regreso al ejecutar la función de sincronización.
- ✓ Ejecutar la función con la siguiente instrucción:

  select sales.sync_customers();

El script correspondiente se puede visualizar en la figura 5.

```
CREATE OR REPLACE FUNCTION sales.sync_customers ()
RETURNS smallint AS
$body$
/* New function body */

declare
  row record;
  cont smallint;

Begin
    cont=1;
    for row in
      select cust.customerid, cust.personid, cust.storeid, cust.territoryid, cust.modifieddate
      from
        dblink(
        'dbname=Adventureworks host=172.20.11.60 user=syncadmin password=passw0rd',
        'select customerid, personid, storeid, territoryid, modifieddate from sales.customer') as cust(customerid integer, personid integer, storeid integer, territoryid integer, modifieddate timestamp)
    loop

        insert into sales.customer (customerid, personid, storeid, territoryid, modifieddate)
        values (row.customerid, row.personid, row.storeid, row.territoryid, row.modifieddate);
        RAISE NOTICE 'INSERT Registro % % % % % %', cont, row.customerid, row.personid, row.storeid, row.territoryid, row.modifieddate;
        cont = cont + 1;

    end loop;
    return cont;
end
$body$
LANGUAGE 'plpgsql';
```

Figura 5 Script para la sincronización de la tabla customer.

El procedimiento general para la sincronización de la tabla "salesorderheader" consiste en lo siguiente:

- ✓ Crear una función de nombre sync_ salesorderheader dentro del esquema sales.

- ✓ Realizar la conexión a la base de datos Adventureworks ubicada en el servidor de producción.
- ✓ Seleccionar todos los registros de la tabla sales.salesorderheader ubicada en el servidor de producción.
- ✓ Se sincronizan las siguientes columnas: *salesorderid, revisionnumber, orderdate, duedate, shipdate, status, onlineorderflag, purchaseordernumber, accountnumber, customerid, salespersonid, territoryid, billtoaddressid, shiptoaddressid, shipmethodid, creditcardid, creditcardapprovalcode, currencyrateid, subtotal, taxamt, freight, totaldue, comment, modifieddate.*
- ✓ Recorrer los registros uno por uno e insertarlos en la tabla del mismo nombre, pero ahora en el servidor de ensayo.
- ✓ Llevar un contador de registros para proporcionarlo como valor de regreso al ejecutar la función de sincronización.
- ✓ Ejecutar la función con la siguiente instrucción:
  select sales.sync_salesorderheader();

El script correspondiente se puede visualizar en la figura 6.

```
CREATE OR REPLACE FUNCTION sales.sync_salesorderheader ()
RETURNS smallint AS
$body$
/* New function body */

declare
  row record;
  cont smallint;

Begin
    cont=1;
    for row in
      select  custord.salesorderid, custord.revisionnumber, custord.orderdate, custord.duedate, custord.shipdate, custord.status, custord.onlineorderflag, custord.purchaseordernumber, custord.accountnumber, custord.customerid, cus
      from
        dblink(
        'dbname=Adventureworks host=172.20.11.60 user=syncadmin password=mcpassvd',
        'select salesorderid, revisionnumber, orderdate, duedate, shipdate, status, onlineorderflag, purchaseordernumber, accountnumber, customerid, salespersonid, territoryid, billtoaddressid, shiptoaddressid, shipmethodid, credi
        as custord(salesorderid integer, revisionnumber smallint, orderdate timestamp, duedate timestamp, shipdate timestamp, status smallint, onlineorderflag boolean, purchaseordernumber varchar, accountnumber varchar, customerid
    Loop

        insert into sales.salesorderheader (salesorderid, revisionnumber, orderdate, duedate, shipdate, status, onlineorderflag, purchaseordernumber, accountnumber, customerid, salespersonid, territoryid, billtoaddressid, shipmeth
        values (row.salesorderid, row.revisionnumber, row.orderdate, row.duedate, row.shipdate, row.status, row.onlineorderflag, row.purchaseordernumber, row.accountnumber, row.customerid, row.salespersonid, row.territoryid, row.b
        RAISE NOTICE 'INSERT Registro % % % % % %', cont, row.salesorderid, row.revisionnumber, row.orderdate, row.duedate, row.shipdate;
        cont = cont + 1;

    end loop;
    return cont;
end
$body$
LANGUAGE 'plpgsql';
```

Figura 6 Script para la sincronización de la tabla salesorderheader.

3. Resultados

Los resultados obtenidos como parte de esta investigación fueron lograr realizar pruebas de nuevas características y funcionalidades de un software en un ambiente lo más real posible, con información que se utiliza en un sistema en producción. De esta manera, los clientes potenciales del software en desarrollo tienen la posibilidad de probarlo no con información aleatoria o ficticia, sino con información utilizada por otros clientes o utilizada por ellos mismos, pero en el entorno real, donde actualmente funciona el software día a día.

Así también se obtuvieron estadísticos de volúmenes de información migrada y sincronizada entre bases de datos como pueden ser: cantidad de tablas, totales de registros por tabla y tamaño de la información. En las tablas 1 y 2 se muestra datos estadísticos de sincronización entre las bases de datos.

Tabla 1 Resultados globales de migración.

Total esquemas sincronizados	Total tablas sincronizadas	Total registros migrados	Tamaño de la información
5	23	307760	88.9 MB

Como consecuencia también se generaron scripts con códigos que permiten adaptarse a diferentes esquemas y arquitecturas de desarrollo de software para que puedan ser reutilizados y aprovechados de la mejor manera. En total se generaron 23 scripts, uno por cada tabla de la base de datos. Si bien estos scripts se pueden optimizar haciendo las llamadas por esquema, lo cual correspondería a generar cinco scripts más en los cuales se manden llamar los correspondientes a las tablas contenidas.

4. Discusión

Realizar pruebas de nuevas funcionalidades en un software que se está desarrollado es parte vital para el éxito de éste, se necesita garantizar el buen funcionamiento y que cumple con los requerimientos establecidos por el cliente. Existen varias formas de llevar a cabo esta actividad, en esta investigación se expuso una de ellas a través del uso del módulo dblink para lograr comunicar dos

bases de datos que a la postre utilizará el software que se esté desarrollando y así lograr el objetivo.

Tabla 2 Resultados con total de registros migrados por esquema de base de datos.

Esquema	Tablas/Registros	
humanresources	department	16
	employee	290
	employeedepartmenthistory	296
	employeepayhistory	316
	Total	**918**
person	address	19614
	person	19972
	personphone	19972
	emailaddress	19972
	Total	79530
production	product	504
	productdescription	762
	productinventory	1069
	Total	**2335**
purchasing	productvendor	460
	purchaseorderdetail	8845
	purchaseorderheader	4012
	vendor	104
	Total	**13421**
sales	creditcard	19118
	customer	19820
	personcreditcard	19118
	salesorderdetail	121317
	salesorderheader	31465
	salesperson	17
	store	701
	Total	**211556**

Algunas de las ventajas de utilizar este método es la velocidad de transmisión de datos ya que no existe ningún intermediario durante la comunicación de las bases de datos, otra ventaja es la rapidez con que se instala y configura la infraestructura necesaria, ya que si se hiciera de otra forma, por ejemplo, con el desarrollo de un segundo software especial para la migración de información o tal vez el desarrollo de servicios web, llevaría mucho más tiempo y consumiría otros recursos económicos y humanos.

5. Bibliografía y Referencias

[1] GitHub Inc. (2017). Adventure Works-for-Postgres: https://goo.gl/SrNk4U.

[2] Jeffer Ochoa. (2015). Tipos de servidores y entornos: https://goo.gl/Mec62s.

[3] Raphael Hertzog and Roland Mas. (2013). The Debian Administrator's Handbook : Creative Commons Licence, GNU General Public Licence.

[4] The PostgreSQL Global Development Group. (2010). PostgreSQL 9.6 Documentation. EU A.: University of California.

[5] PostgreSQL Web Site (2017). Dblink: https://goo.gl/Cc57qa.