



MANUAL TÉCNICO

ETAB

Esta página se ha dejado vacía a propósito

Índice de contenidos

| | |
|--|-----------|
| Capítulo 1 Tecnologías utilizadas..... | 7 |
| Capítulo 2 Tablero eTAB | 11 |
| 2.1 Instalacion del Sistema Integrado de Información Gerencial | 11 |
| 2.2 Requerimientos..... | 11 |
| 2.3 Pasos..... | 11 |
| 2.4 1. Instalación de Symfony2..... | 11 |
| 2.5 Configuración..... | 12 |
| 2.6 2. Instalación de Postgres | 14 |
| 2.7 3. Instalación de RabbitMQ..... | 15 |
| 2.8 4. Instalación de Servidor OLAP | 16 |
| Capítulo 3 Esquema general de la Aplicacion..... | 17 |
| 3.1 Diagrama Entidad Relacion..... | 18 |
| 3.2 Diccionario de Datos | 18 |
| Capítulo 4 Gestión de Cubos OLAP | 45 |
| 4.1 Funciones Auxiliares de PostgreSQL | 51 |

Esta página se ha dejado vacía a propósito

Introducción

El presente manual técnico describe cada uno de los componentes del sistema eTAB y los pasos necesarios para instalarlo.

El sistema de información eTAB es parte de la iniciativa Salud Mesoamérica 2015 (SM2015). Esta es una iniciativa cuyo fin es reducir las inequidades en salud que están afectando al 20 por ciento mas pobre de la población en Centro America y Mexico. Esta iniciativa también tiene como objetivo apoyar los esfuerzos de los gobiernos de la región para alcanzar los Objetivos del Milenio.

Salud Mesoamérica 2015, pone especial atención a la áreas de salud reproductiva, nutrición maternal y neonatal, inmunización, y la prevención y control del dengue y la malaria. Para este fin Salud Mesoamérica 2015trabajara en conjunto con los ministerios de salud de la región y el Sistema de Salud Publica Mesoamericano. Este proyecto es parte de la plataforma de integración regional conocido como Proyecto Mesoamerica.

Los resultados esperados de incluyen una reducción significativa en las tasa de mortalidad infantil para niños de menos de cinco años. Esta iniciativa también esta busca reducir la malnutrición crónica en la niñez y las mujeres embarazadas. Estos cambios son críticos para mejorar las estadísticas sobre partos y para ofrecer mejores condiciones para el crecimiento del recién nacido. A su vez esta iniciativa busca tener un efecto directo en comunidades pobres sobre la cobertura y calidad de vacunas, control pre y post natal y acceso a planificación familiar entre otros servicios.

Salud Mesoamérica 2015 espera generar conocimiento de relevancia global sobre como aumentar asistencia en salud de bajo costo en comunidades pobres. Para este fin el sistema de información eTAB permite analizar y dar seguimiento a los indicadores en salud con los que trabaja este proyecto.

Esta página se ha dejado vacía a propósito

Capítulo 1

Tecnologías utilizadas

El Tablero eTAB es un sistema que funciona como un servicio Web disponible para que dependencias del sistema de salud suban sus datos para poder analizarlos, generar gráficas y reportes.

La aplicación cuenta con un módulo para efectuar la extracción, transformación y carga de datos (ETL) desde diferentes fuentes. Estos datos son agregados y almacenados en una base de datos relacional (OLTP). Los datos están organizados por catálogos de referencia e Indicadores medibles. Los usuarios del sistema pueden administrar estos indicadores y catálogos y todos sus tributos usando las herramientas que brinda el sistema. Para efectuar consultas en línea los datos son agregados dentro de tablas optimizadas para el análisis en línea (OLAP). Las tablas para análisis son actualizadas periódicamente usando procedimientos almacenados de PostgreSQL.

La gestión de consultas a las tablas de análisis OLAP se realiza por medio de un servidor dedicado. La interacción entre el servidor OLAP y el resto de la aplicación se realiza por medio de consultas AJAX. El resultado de las consultas al servidor OLAP, es procesado usando JQuery y graficado usando la librería de gráficos D3.

Todo el software utilizado para creación del SIIG/eTAB son paquetes de software libre. Estos incluyen:

- GitHub: Gestor de control de versiones de código fuente
- Apache: Servidor de páginas web
- PostgreSQL: Gestor de bases de Datos
- Symfony: Entorno de desarrollo para PHP
- PHP: Lenguaje de desarrollo de la Aplicación eTAB

- Python: Lenguaje de desarrollo del servidor OLAP
- Cubes: Servidor OLAP desarrollado en Python
- D3.js: Librería para la generación de gráficos
- JQuery: Lenguaje para interfaces de usuario
- RabbitMQ: Servidor de Mensajería
- EasyBook: Generador de documentos en formato PDF

1.0.1 Gestor de base de datos

[PostgreSQL] (<http://www.postgresql.org/>)

Versión 9.2 Actualmente le sistema únicamente puede utilizar PostgreSQL por la siguiente razón: La aplicación debe proveer la capacidad de analizar datos para cualquier indicador. Cada indicador esta construido con varios datos y relacionados por una formula almacenada en el sistema. Es posible crear una tabla por cada grupo de datos, con la limitante de que es necesario conocer el dato antes de guardarlo, lo cual no es sostenible a futuro.

La base de datos necesita guardar datos sin conocer de antemano sus características. Esto se logra usando un esquema de datos generico EAV (entidad-atributo-valor). El manejo de esquemas EAV es implementado de diferentes formas para diferentes gestores de bases de datos. El Tablero eTAB usa la implementación de Postgres la cual crea un tipo especial de dato llamado HSTORE. Además, el modulo de cubos OLAP utiliza la función CROSSTAB de Postgres para transponer grupos de datos. El tipo de campo HSTORE y la función CROSSTAB no existen en otros gestores de bases de datos por lo cual no seria posible instalar esta aplicación usando un gestor de base de datos que no sea PostgreSQL.

Para utilizar el tipo de datos HSTORE y la función CROSSTAB es necesario instalar el paquete contrib de postgresql y luego crear las extensiones correspondientes usando:

```
~#postgres> CREATE EXTENSION hstore ;
```

```
~#postgres> CREATE EXTENSION tablefunc ;
```

1.0.2 Servidor Web

[Apache2] (<http://www.apache.org>)

Apache es un servidor Web de código abierto, se ha realizado sobre Apache versión 2.2

1.0.3 Framework de desarrollo/Servidor

[PHP] (<http://www.php.net>)

[Symfony](<http://symfony.com/>)

[GitHub](<https://github.com/>)

El lenguaje que se ha utilizado es PHP 5.3.18 dentro de una estructura de desarrollo MVC manejada Symfony versión 2.4 Cada miembro del equipo de desarrollo usa un aplicativo diferente para escribir/modificar el código fuente. Los mas populares popular es Netbeans(version libre para PHP) y Nano. Para manejar cambios y mejoras al código fuente se uso Github. La totalidad del código fuente esta disponible en <https://github.com/erodriguez-minsal/SIIG>

1.0.4 Framework JavaScript

jQuery (<http://jquery.com/>) versión 1.8.3 junto a jQuery UI (<http://jqueryui.com/>)

1.0.5 Framework para interfaces de usuario

Bootstrap (<http://twitter.github.com/bootstrap/>)

Bootstrap es un framework que hace HTML, CSS y Javascript simple y flexible para componentes e interacciones de interfaz de usuarios populares.

1.0.6 Librería para gráficos

D3 (<http://d3js.org/>)

Antes conocida como Protovis, D3 es una biblioteca de JavaScript para manipular documentos basados en datos. D3 ayuda dar vida a los datos usando HTML, SVG y CSS. D3 enfatiza los estándares Web ofreciendo todas las capacidades de los navegadores modernos sin ligarse a una estructura propietaria. A diferencia de otras librerías, D3 no crea imágenes, sino que interactúa la pagina para crear los gráficos usando elementos de HTML5 como Canvas y SVG.

1.0.7 Mensajería

RabbitMQ (<http://www.rabbitmq.com/>)

La carga de datos se apoya de las librerías de este paquete para crear una ‘lista de espera’ para evitar que el servidor se sature al recibir demasiadas peticiones simultaneas.

1.0.8 Servidor de Cubos OLAP

Python OLAP Cubes (<http://packages.python.org/cubes/>)

El servidor de cubos OLAP es un proyecto de código abierto desarrollado usando Python 2.7, actualmente esta en la versión 0.10

1.0.9 Documentación

La mayoría de la documentación ha sido escrita en formato markdown y se ha utilizado easybook (<http://easybook-project.org/>) para la gen

Capítulo 2

Tablero eTAB

2.1 Instalacion del Sistema Integrado de Información Gerencial

2.2 Requerimientos

- Servidor Web
- Gestor de base de datos
- PHP 5.3.8+

2.3 Pasos

1. Instalación de Symfony2
2. Instalación de Postgres
3. Instalación RabbitMQ
4. Instalación Servidor OLAP

2.4 1. Instalación de Symfony2

2.4.1 Instalación de los requerimientos desde un servidor Debian

Es muy importante poner atención al indicador "#" significa que el comando debe ser ejecutado como usuario root y "\$" que debe ser ejecutado como un usuario normal

```
# apt-get update
# apt-get install php5 php5-pgsql php5-sqlite sqlite php5-xdebug
php-apc php5-cli php5-xsl php5-intl php5-mcrypt apache2 postgresql
acl git-core curl postgresql-contrib
```

2.4.2 Obtener el código fuente

Puedes descargarlo desde: <https://github.com/erodriguez-minsal/SIIG/tarball/master> o clonar el repositorio

```
$ git clone https://github.com/erodriguez-minsal/SIIG.git siig
```

A partir de este punto todos los comandos se deben ejecutar dentro de la carpeta en que se ha descargado el código fuente

2.4.3 Instalar composer

Composer (<http://getcomposer.org/>) Es una librería de PHP para el manejo de dependencias. Para instalarlo, dentro de la carpeta donde descargaste el código fuente se debe ejecutar:

```
$ curl -s https://getcomposer.org/installer | php
```

2.4.4 Instalar todas las librerías necesarias

```
$ php composer.phar install
```

2.5 Configuración

2.5.1 Servidor web

Esto es para una instalación de prueba en una máquina local, la instalación real en un servidor el administrador de servicios deberá realizar esta configuración con los parámetros más adecuados: ip, dominio, configuración en el DNS, etc.

2.5.1.1 Configurar un VirtualHost

Creamos el archivo para la definición del VirtualHost

```
# nano /etc/apache2/sites-available/siig.localhost
```

El contenido será similar a esto:

```
<VirtualHost 127.0.0.7>

    ServerName siig.localhost
    DocumentRoot /ruta_al_directorio_descargado/web
```

```
<Directory /ruta_al_directorio_descargado/web >
    Options Indexes FollowSymLinks MultiViews
    AllowOverride All
    Order allow,deny
    allow from all
</Directory>

ErrorLog ${APACHE_LOG_DIR}/siig-error.localhost.log
# Possible values include: debug, info, notice, warn, error,
crit,
# alert, emerg.
LogLevel warn
CustomLog ${APACHE_LOG_DIR}/siig-access.localhost.log combined
</VirtualHost>
```

En el archivo `/etc/hosts` agregamos la línea

```
127.0.0.7          siig.localhost
```

Habilitamos el `VirtualHost`

```
# a2ensite siig.localhost
```

También es recomendable activar el módulo `mod_rewrite`

```
# a2enmod rewrite
```

Reiniciar apache

```
# /etc/init.d/apache2 restart
```

2.5.2 Permisos sobre carpetas

Es necesario tener soporte para ACL (<https://help.ubuntu.com/community/FilePermissionsACLs>) en la partición en que está el proyecto y luego ejecutar

```
# setfacl -R -m u:www-data:rwx -m u:`whoami`:rwx app/cache app/
logs web/uploads
# setfacl -dR -m u:www-data:rwx -m u:`whoami`:rwx app/cache app/
logs web/uploads
```

2.5.3 Verificar la configuración

Entra a la siguiente dirección desde el navegador `http://siig.localhost/config.php` Si aparece algún error debe ser corregido antes de continuar

2.5.4 Configuración de la conexión

editar el archivo `app/config/parameters.yml` y colocar los valores correctos para las variables siguientes:

- `database_host: localhost`
- `database_port: null`
- `database_name: nombre_base_datos`
- `database_user: nombre_usuario_base_datos`
- `database_password: clave_usuario`

2.6 2. Instalación de Postgres

2.6.1 Crear la base de datos

```
$ app/console doctrine:database:create
$ app/console doctrine:schema:update --force
```

2.6.2 Cargar datos iniciales

```
$ app/console doctrine:fixtures:load
```

2.6.3 Crear un usuario administrador

```
$ app/console fos:user:create --super-admin
```

2.6.4 Instalación de HStore

HStore (<http://www.postgresql.org/docs/9.1/static/hstore.html>) es un tipo especial de campo de PostgreSQL

- Ejecutar dentro de la base de datos, con el usuario postgres

```
create extension hstore;
```

- Crear la tabla especial que no se manejará con el ORM, hacerlo con el usuario dueño de la base de datos
- (no con el usuario postgres, a menos que este mismo sea el dueño de la base de datos)

```
CREATE TABLE fila_origen_dato(  
    id serial,  
    id_origen_dato integer,  
    datos hstore,  
  
    PRIMARY KEY (id),  
    FOREIGN KEY (id_origen_dato) REFERENCES origen_datos(id) on update  
    CASCADE on delete RESTRICT  
  
);
```

2.7 3. Instalación de RabbitMQ

2.7.1 Instalación de RabbitMQ

RabbitMQ (<http://www.rabbitmq.com/>) es un sistema de mensajería empresarial completo y altamente confiable basado en el estándar AMQP. Charla sobre RabbitMQ (<http://www.symfony.es/noticias/2011/07/06/desymfony-2011-reduciendo-el-acoplamiento-entre-aplicaciones-con-rabbitmq/>). En este proyecto será utilizado para la carga masiva de datos y así evitar cuelgues o saturación del servidor.

- Agregar el repositorio

```
# sh -c 'echo "deb http://www.rabbitmq.com/debian/ testing main"  
>> /etc/apt/sources.list'
```

- Agregar la clave pública

```
# wget http://www.rabbitmq.com/rabbitmq-signing-key-public.asc  
# apt-key add rabbitmq-signing-key-public.asc
```

- Ejecutar

```
# apt-get update
```

- Instalar el paquete

```
# apt-get install rabbitmq-server
```

- Verificar que el servicio de rabbitmq esté corriendo

```
# /etc/init.d/rabbitmq-server start
```

- Iniciar las colas

```
$ src/MINSAL/IndicadoresBundle/Util/iniciar_colas.sh
```

Pueden aparecer mensajes de aviso como `"/usr/bin/nohup: redirecting stderr to stdout"` solo debemos presionar ENTER

- Habilitar la interfaz web de administración

```
# rabbitmq-plugins enable rabbitmq_management
# /etc/init.d/rabbitmq-server restart
```

- Cargar la interfaz web: entrar a la dirección `http://server_name:55672/mgmt/` El usuario por defecto es **guest** y la clave **guest**

2.8 4. Instalación de Servidor OLAP

En Debian squeeze con repositorio 'testing':

```
~#apt-get install python2.7 python-sqlalchemy python-werkzeug python-sqlite python-psycpg2 python-pip
```

Finalmente se debe usar un gestor de paquetes de Python para instalar el paquete del servidor OLAP 'Cubes'. La siguiente línea utiliza el gestor 'pip'.

```
~# pip install cubes Desde la Carpeta siig/src/MINSAL/cubos, ejecutar:
```

```
~# slicer serve slicer.ini
```

Este comando ejecuta el servidor OLAP y muestra la salida en pantalla. La documentación completa de este servidor dentro de la aplicación está disponible en la sección "Estión y Análisis de Cubos OLAP"

2.8.1 Cargar la aplicación

En este punto estamos listos para cargar la aplicación desde: `http://siig.localhost/app_dev.php`

Capítulo 3

Esquema general de la Aplicación

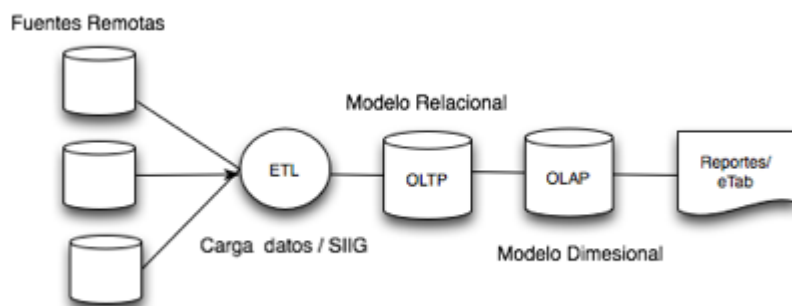


Figura 3.1 Esquema de la aplicación

Los datos que maneja el sistema provienen de distintas fuentes y son de una naturaleza tal que es necesario utilizar el modelo de base datos sin esquema/genérico EAV. Las tabla EAV (Fila_origen_dato) y demás tablas auxiliares son parte del almacenamiento de datos transaccional (OLTP) de la aplicación. Esto facilita el manejo de datos de cualquier indicador sin importar sus propiedades. Los cubos de análisis multidimensional (OLAP) son generados usando estos valores genéricos y están descritos en la sección de Gestión de Cubos OLAP. Las tablas de los cubos OLAP usan un esquema de estrella mientras que las tablas del almacenamiento OLTP usan un modelo relacional. El siguiente Diccionario de Datos y Diagrama ER describen la estructura del almacenamiento transaccional (OLTP) de la Aplicación.

- [ficha_tecnica_campo](#) (página 26)
- [ficha_tecnica_periodicidad](#) (página 27)
- [ficha_tecnica_presentacion](#) (página 28)
- [ficha_tecnica_variable_dato](#) (página 28)
- [fila_origen_dato](#) (página 29)
- [fos_user_group](#) (página 30)
- [fos_user_user](#) (página 30)
- [fos_user_user_group](#) (página 32)
- [fuente_dato](#) (página 33)
- [fusion_origenes_datos](#) (página 34)
- [indicador_alertas](#) (página 34)
- [motor_bd](#) (página 35)
- [origen_datos](#) (página 36)
- [origen_datos_fusiones](#) (página 37)
- [periodos](#) (página 37)
- [presentacion](#) (página 38)
- [responsable_dato](#) (página 39)
- [responsable_indicador](#) (página 39)
- [significado_campo](#) (página 40)
- [tipo_campo](#) (página 41)
- [usuario_indicadores_favoritos](#) (página 42)
- [variable_dato](#) (página 42)

1. alerta ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| codigo | codigo | VARCHAR(30) | | NOT NULL |
| color | color | VARCHAR(50) | | NOT NULL |

Esta tabla es usada por:

<

ul>

- [indicador_alertas \(página 34\)](#) hace referencia la campo (id)

2. campo ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|---|--------------------------|---------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| id_origen_datos (FK (página 36)) | id_origen_datos | BIGINT | | NOT NULL |
| id_tipo_campo (FK (página 41)) | id_tipo_campo | INTEGER | | NOT NULL |

| | | | | |
|--|----------------------|--------------|--|----------|
| id_significado_campo (FK (página 40)) | id_significado_campo | INTEGER | | NOT NULL |
| nombre | nombre | VARCHAR(100) | | NOT NULL |
| descripcion | descripcion | CLOB | | |

Esta tabla depende de:

- [origen_datos \(página 36\)](#) por medio de (id_origen_datos)
- [tipo_campo \(página 41\)](#) por medio de (id_tipo_campo)
- [significado_campo \(página 40\)](#) por medio de (id_significado_campo)

Esta tabla es usada por:

- [ficha_tecnica_campo \(página 26\)](#) hace referencia la campo (id)

3. clasificacion_nivel ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| codigo | codigo | VARCHAR(15) | | NOT NULL |
| descripcion | descripcion | VARCHAR(50) | | NOT NULL |
| comentario | comentario | CLOB | | |

Esta tabla es usada por:

- [ficha_tecnica \(página 25\)](#) hace referencia la campo (id)

4. clasificacion_privacidad ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| codigo | codigo | VARCHAR(15) | | NOT NULL |
| descripcion | descripcion | VARCHAR(50) | | NOT NULL |
| comentario | comentario | CLOB | | |

Esta tabla es usada por:

- [ficha_tecnica \(página 25\)](#) hace referencia la campo (id)

5. clasificacion_tecnica ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| codigo | codigo | VARCHAR(15) | | NOT NULL |
| descripcion | descripcion | VARCHAR(50) | | NOT NULL |
| comentario | comentario | CLOB | | |

Esta tabla es usada por:

- [ficha_tecnica \(página 25\)](#) hace referencia la campo (id)

6. clasificacion_uso ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| codigo | codigo | VARCHAR(15) | | NOT NULL |
| descripcion | descripcion | VARCHAR(50) | | NOT NULL |
| comentario | comentario | CLOB | | |

Esta tabla es usada por:

- ficha_tecnica (página 25) hace referencia la campo (id)

7. conexion ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|------------------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| id_motor (FK (<u>página 35</u>)) | id_motor | INTEGER | | NOT NULL |
| nombre_conexion | nombre_conexion | VARCHAR(100) | | NOT NULL |
| comentario | comentario | CLOB | | |
| ip | ip | VARCHAR(15) | | NOT NULL |
| usuario | usuario | VARCHAR(25) | | NOT NULL |
| clave | clave | VARCHAR(150) | | NOT NULL |
| nombre_base_datos | nombre_base_datos | VARCHAR(50) | | NOT NULL |
| puerto | puerto | VARCHAR(5) | | |
| instancia | instancia | VARCHAR(50) | | |

Esta tabla depende de:

- motor_bd (página 35) por medio de (id_motor)

Esta tabla es usada por:

- origen_datos (página 36) hace referencia la campo (id)

8. ficha_tecnica ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|--|-----------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| id_clasificacion_nivel (FK (página 21)) | id_clasificacion_nivel | INTEGER | | NOT NULL |
| id_clasificacion_privacidad (FK (página 22)) | id_clasificacion_privacidad | INTEGER | | NOT NULL |
| id_clasificacion_tecnica (FK (página 23)) | id_clasificacion_tecnica | INTEGER | | NOT NULL |
| id_clasificacion_uso (FK (página 23)) | id_clasificacion_uso | INTEGER | | NOT NULL |
| id_responsable_indicador (FK (página 39)) | id_responsable_indicador | INTEGER | | NOT NULL |
| nombre | nombre | VARCHAR(150) | | NOT NULL |
| tema | tema | CLOB | | NOT NULL |
| concepto | concepto | CLOB | | |
| objetivo | objetivo | CLOB | | |
| uso | uso | CLOB | | |
| definicion_operativa | definicion_operativa | CLOB | | |
| unidad_medida | unidad_medida | VARCHAR(50) | | NOT NULL |
| formula | formula | VARCHAR(300) | | NOT NULL |
| observacion | observacion | CLOB | | |
| campos_indicador | campos_indicador | CLOB | | |
| confiabilidad | confiabilidad | INTEGER | | |
| updated_at | updated_at | TIMESTAMP | | |
| id_periodo (FK (página 37)) | id_periodo | INTEGER | | NOT NULL |
| ultima_lectura | ultima_lectura | TIMESTAMP | | |

Esta tabla depende de:

- [periodos \(página 37\)](#) por medio de (id_periodo)
- [clasificacion_privacidad \(página 22\)](#) por medio de (id_clasificacion_privacidad)
- [clasificacion_tecnica \(página 23\)](#) por medio de (id_clasificacion_tecnica)
- [clasificacion_uso \(página 23\)](#) por medio de (id_clasificacion_uso)
- [clasificacion_nivel \(página 21\)](#) por medio de (id_clasificacion_nivel)
- [responsable_indicador \(página 39\)](#) por medio de (id_responsable_indicador)

Esta tabla es usada por:

- [ficha_tecnica_periodicidad \(página 27\)](#) hace referencia la campo (id)
- [ficha_tecnica_campo \(página 26\)](#) hace referencia la campo (id)
- [ficha_tecnica_presentacion \(página 28\)](#) hace referencia la campo (id)
- [indicador_alertas \(página 34\)](#) hace referencia la campo (id)
- [ficha_tecnica_variable_dato \(página 28\)](#) hace referencia la campo (id)
- [usuario_indicadores_favoritos \(página 42\)](#) hace referencia la campo (id)

9. ficha_tecnica_campo ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|---|--------------------------|---------|----|----------|
| id_ficha_tecnica (PK) (FK (página 25)) | id_ficha_tecnica | INTEGER | PK | NOT NULL |

| | | | | |
|--------------------------------|----------|---------|----|----------|
| id_campo (PK) (FK (página 20)) | id_campo | INTEGER | PK | NOT NULL |
|--------------------------------|----------|---------|----|----------|

Esta tabla depende de:

- campo (página 20) por medio de (id_campo)
- ficha_tecnica (página 25) por medio de (id_ficha_tecnica)

10. ficha_tecnica_periodicidad ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|--|--------------------------|---------|----|----------|
| id_ficha_tecnica (PK) (FK (página 25)) | id_ficha_tecnica | INTEGER | PK | NOT NULL |
| id_periodicidad (PK) (FK (página 37)) | id_periodicidad | INTEGER | PK | NOT NULL |

Esta tabla depende de:

- periodos (página 37) por medio de (id_periodicidad)
- ficha_tecnica (página 25) por medio de (id_ficha_tecnica)

11. ficha_tecnica_presentacion ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|---|--------------------------|---------|----|----------|
| id_ficha_tecnica (PK) (FK (página 25)) | id_ficha_tecnica | INTEGER | PK | NOT NULL |
| id_presentacion (PK) (FK (página 38)) | id_presentacion | INTEGER | PK | NOT NULL |

Esta tabla depende de:

- [ficha_tecnica \(página 25\)](#) por medio de (id_ficha_tecnica)
- [presentacion \(página 38\)](#) por medio de (id_presentacion)

12. ficha_tecnica_variable_dato ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|------|----|----------|
|-------------------------|--------------------------|------|----|----------|

| | | | | |
|--|------------------|---------|----|----------|
| id_ficha_tecnica (PK) (FK (página 25)) | id_ficha_tecnica | INTEGER | PK | NOT NULL |
| id_variable_dato (PK) (FK (página 42)) | id_variable_dato | INTEGER | PK | NOT NULL |

Esta tabla depende de:

- [ficha_tecnica \(página 25\)](#) por medio de (id_ficha_tecnica)
- [variable_dato \(página 42\)](#) por medio de (id_variable_dato)

13. fila_origen_dato ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|---------------------------------|--------------------------|---------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| id_origen_dato (FK (página 36)) | id_origen_dato | BIGINT | | NOT NULL |
| datos | datos | [1111] | | |

Esta tabla depende de:

- [origen_datos \(página 36\)](#) por medio de (id_origen_dato)

14. fos_user_group ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| name | name | VARCHAR(255) | | NOT NULL |
| roles | roles | CLOB | | NOT NULL |

(DC2Tipo:array)

Esta tabla es usada por:

- [fos_user_user_group \(página 32\)](#) hace referencia la campo (id)

15. fos_user_user ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| username | username | VARCHAR(255) | | NOT NULL |
| username_canonical | username_canonical | VARCHAR(255) | | NOT NULL |
| email | email | VARCHAR(255) | | NOT NULL |
| email_canonical | email_canonical | VARCHAR(255) | | NOT NULL |
| enabled | enabled | BOOLEAN | | NOT NULL |

| | | | | |
|-----------------------|-----------------------|--------------|--|----------|
| salt | salt | VARCHAR(255) | | NOT NULL |
| password | password | VARCHAR(255) | | NOT NULL |
| last_login | last_login | TIMESTAMP | | |
| locked | locked | BOOLEAN | | NOT NULL |
| expired | expired | BOOLEAN | | NOT NULL |
| expires_at | expires_at | TIMESTAMP | | |
| confirmation_token | confirmation_token | VARCHAR(255) | | |
| password_requested_at | password_requested_at | TIMESTAMP | | |
| roles | roles | CLOB | | NOT NULL |
| credentials_expired | credentials_expired | BOOLEAN | | NOT NULL |
| credentials_expire_at | credentials_expire_at | TIMESTAMP | | |
| created_at | created_at | TIMESTAMP | | NOT NULL |
| updated_at | updated_at | TIMESTAMP | | NOT NULL |
| date_of_birth | date_of_birth | TIMESTAMP | | |
| firstname | firstname | VARCHAR(64) | | |
| lastname | lastname | VARCHAR(64) | | |
| website | website | VARCHAR(64) | | |
| biography | biography | VARCHAR(255) | | |
| gender | gender | VARCHAR(1) | | |
| locale | locale | VARCHAR(8) | | |
| timezone | timezone | VARCHAR(64) | | |
| phone | phone | VARCHAR(64) | | |
| facebook_uid | facebook_uid | VARCHAR(255) | | |
| facebook_name | facebook_name | VARCHAR(255) | | |
| facebook_data | facebook_data | CLOB | | |
| twitter_uid | twitter_uid | VARCHAR(255) | | |

| | | | | |
|---------------|---------------|--------------|--|--|
| twitter_name | twitter_name | VARCHAR(255) | | |
| twitter_data | twitter_data | CLOB | | |
| gplus_uid | gplus_uid | VARCHAR(255) | | |
| gplus_name | gplus_name | VARCHAR(255) | | |
| gplus_data | gplus_data | CLOB | | |
| token | token | VARCHAR(255) | | |
| two_step_code | two_step_code | VARCHAR(255) | | |

Esta tabla es usada por:

- [fos_user_user_group](#) (página 32) hace referencia la campo (id)
- [usuario_indicadores_favoritos](#) (página 42) hace referencia la campo (id)

16. fos_user_user_group ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|--------------------------------|--------------------------|---------|----|----------|
| user_id (PK) (FK (página 30)) | user_id | INTEGER | PK | NOT NULL |
| group_id (PK) (FK (página 30)) | group_id | INTEGER | PK | NOT NULL |

Esta tabla depende de:

- `fos_user_group` (página 30) por medio de (`group_id`)
- `fos_user_user` (página 30) por medio de (`user_id`)

17. fuente_dato ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| establecimiento | establecimiento | VARCHAR(100) | | NOT NULL |
| contacto | contacto | VARCHAR(100) | | NOT NULL |
| correo | correo | VARCHAR(50) | | NOT NULL |
| telefono | telefono | VARCHAR(15) | | NOT NULL |
| cargo | cargo | VARCHAR(50) | | NOT NULL |

Esta tabla es usada por:

- `variable_dato` (página 42) hace referencia la campo (`id`)

18. fusion_origenes_datos ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|--|---------------------------|---------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| id_origen_datos (FK (página 36)) | id_origen_datos | BIGINT | | NOT NULL |
| id_origen_datos_fusionado (FK (página 36)) | id_origen_datos_fusionado | BIGINT | | NOT NULL |
| campos | campos | CLOB | | |

Esta tabla depende de:

- [origen_datos \(página 36\)](#) por medio de (id_origen_datos)
- [origen_datos \(página 36\)](#) por medio de (id_origen_datos_fusionado)

19. indicador_alertas ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|---------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |

| | | | | |
|----------------------------------|-----------------|---------|--|----------|
| id_color_alerta (FK (página 20)) | id_color_alerta | INTEGER | | NOT NULL |
| id_indicador (FK (página 25)) | id_indicador | INTEGER | | NOT NULL |
| limite_inferior | limite_inferior | DOUBLE | | NOT NULL |
| limite_superior | limite_superior | DOUBLE | | NOT NULL |
| comentario | comentario | CLOB | | |

Esta tabla depende de:

- alerta (página 20) por medio de (id_color_alerta)
- ficha_tecnica (página 25) por medio de (id_indicador)

20. motor_bd ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| nombre | nombre | VARCHAR(50) | | NOT NULL |
| codigo | codigo | VARCHAR(20) | | |

Esta tabla es usada por:

- conexion (página 24) hace referencia la campo (id)

21. origen_datos ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|------------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | BIGINT | PK | NOT NULL |
| id_conexion (FK (página 24)) | id_conexion | INTEGER | | NOT NULL |
| nombre | nombre | VARCHAR(100) | | NOT NULL |
| descripcion | descripcion | CLOB | | |
| sentencia_sql | sentencia_sql | CLOB | | |
| archivo_nombre | archivo_nombre | VARCHAR(100) | | |
| es_fusionado | es_fusionado | BOOLEAN | | |
| es_catalogo | es_catalogo | BOOLEAN | | |
| nombre_catalogo | nombre_catalogo | VARCHAR(100) | | |
| campos_fusionados | campos_fusionados | CLOB | | |

Esta tabla depende de:

- [conexion \(página 24\)](#) por medio de (id_conexion)

Esta tabla es usada por:

- [campo \(página 20\)](#) hace referencia la campo (id)
- [origen_datos_fusiones \(página 37\)](#) hace referencia la campo (id)
- [origen_datos_fusiones \(página 37\)](#) hace referencia la campo (id)
- [variable_dato \(página 42\)](#) hace referencia la campo (id)
- [fila_origen_dato \(página 29\)](#) hace referencia la campo (id)
- [fusion_origenes_datos \(página 34\)](#) hace referencia la campo (id)
- [fusion_origenes_datos \(página 34\)](#) hace referencia la campo (id)

22. origen_datos_fusiones ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|--|--------------------------|--------|----|----------|
| id_origen_dato (PK) (FK (página 36)) | id_origen_dato | BIGINT | PK | NOT NULL |
| id_origen_dato_fusionado (PK) (FK (página 36)) | id_origen_dato_fusionado | BIGINT | PK | NOT NULL |

Esta tabla depende de:

- [origen_datos \(página 36\)](#) por medio de (id_origen_dato_fusionado)
- [origen_datos \(página 36\)](#) por medio de (id_origen_dato)

23. periodos ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| descripcion | descripcion | VARCHAR(25) | | NOT NULL |
| codigo | codigo | VARCHAR(7) | | NOT NULL |

| | | | | |
|-----------|-----------|------|--|--|
| sentencia | sentencia | CLOB | | |
|-----------|-----------|------|--|--|

Esta tabla es usada por:

- [ficha_tecnica_periodicidad \(página 27\)](#) hace referencia la campo (id)
- [ficha_tecnica \(página 25\)](#) hace referencia la campo (id)

24. presentacion ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| codigo | codigo | VARCHAR(20) | | NOT NULL |
| descripcion | descripcion | VARCHAR(40) | | NOT NULL |

Esta tabla es usada por:

- [ficha_tecnica_presentacion \(página 28\)](#) hace referencia la campo (id)

25. responsable_dato ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| establecimiento | establecimiento | VARCHAR(100) | | NOT NULL |
| contacto | contacto | VARCHAR(100) | | NOT NULL |
| correo | correo | VARCHAR(50) | | NOT NULL |
| telefono | telefono | VARCHAR(15) | | NOT NULL |
| cargo | cargo | VARCHAR(50) | | NOT NULL |

Esta tabla es usada por:

- [variable_dato \(página 42\)](#) hace referencia la campo (id)

26. responsable_indicador ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|---------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |

| | | | | |
|-----------------|-----------------|--------------|--|----------|
| establecimiento | establecimiento | VARCHAR(100) | | NOT NULL |
| contacto | contacto | VARCHAR(100) | | NOT NULL |
| correo | correo | VARCHAR(50) | | NOT NULL |
| telefono | telefono | VARCHAR(15) | | NOT NULL |
| cargo | cargo | VARCHAR(50) | | NOT NULL |

Esta tabla es usada por:

- [ficha_tecnica \(página 25\)](#) hace referencia la campo (id)

27. significado_campo()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|--------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| descripcion | descripcion | VARCHAR(200) | | NOT NULL |
| codigo | codigo | VARCHAR(40) | | NOT NULL |
| uso_en_catalogo | uso_en_catalogo | BOOLEAN | | |
| catalogo | catalogo | VARCHAR(255) | | |
| nombre_mapa | nombre_mapa | VARCHAR(200) | | |

Esta tabla es usada por:

- [campo \(página 20\)](#) hace referencia la campo (id)

28. tipo_campo ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|-------------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |
| descripcion | descripcion | VARCHAR(50) | | |
| codigo | codigo | VARCHAR(50) | | NOT NULL |

Esta tabla es usada por:

- [campo \(página 20\)](#) hace referencia la campo (id)

29. usuario_indicadores_favoritos ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|------------------------------------|--------------------------|---------|----|----------|
| id_usuario (PK) (FK (página 30)) | id_usuario | INTEGER | PK | NOT NULL |
| id_indicador (PK) (FK (página 25)) | id_indicador | INTEGER | PK | NOT NULL |

Esta tabla depende de:

```
<ul>
  <li> <a href="#ficha_tecnica">ficha_tecnica</a> por medio de
(id_indicador)</li>
  <li> <a href="#fos_user_user">fos_user_user</a> por medio de
(id_usuario)</li>
</ul>
```

29. variable_dato ()

| Nomre Logico de Columna | Nombre Fisico de Columna | Tipo | PK | Nullable |
|-------------------------|--------------------------|---------|----|----------|
| id (PK) | id | INTEGER | PK | NOT NULL |

| | | | | |
|---|---------------------|--------------|--|----------|
| id_fuente_dato (FK (página 33)) | id_fuente_dato | INTEGER | | NOT NULL |
| id_responsable_dato (FK (página 39)) | id_responsable_dato | INTEGER | | NOT NULL |
| id_origen_datos (FK (página 36)) | id_origen_datos | BIGINT | | NOT NULL |
| nombre | nombre | VARCHAR(200) | | NOT NULL |
| confiabilidad | confiabilidad | INTEGER | | |
| iniciales | iniciales | VARCHAR(255) | | NOT NULL |
| comentario | comentario | CLOB | | |
| es_poblacion | es_poblacion | BOOLEAN | | |

Esta tabla depende de:

- [origen_datos \(página 36\)](#) por medio de (id_origen_datos)
- [fuente_dato \(página 33\)](#) por medio de (id_fuente_dato)
- [responsable_dato \(página 39\)](#) por medio de (id_responsable_dato)

Esta tabla es usada por:

- [ficha_tecnica_variable_dato \(página 28\)](#) hace referencia la campo (id)

Esta página se ha dejado vacía a propósito

Capítulo 4

Gestión de Cubos OLAP

4.0.1 Introducción

La cantidad de reportes que pueden generarse depende de los catálogos que se usen en cada indicador. Al interior de una estructura para reportes (cubo OLAP), cada catalogo se convierte en dimensión. Así por ejemplo el catalogo municipio se convierte en una dimensión dentro de la estructura de reportes y posibilita hacer búsquedas usando cualquier campo que exista dentro de este catálogo. Por ejemplo: Si el catalogo municipio tiene los campos municipio, departamento y región esto nos permitiría hacer las siguientes consultas: numero de casos por municipio, número de casos por departamento, número de casos por región.

La ventaja principal de usar un gestor de cubos OLAP es aislar la lógica de las búsquedas para analizar los datos. De esta forma el sistema se enfoca en presentar al usuario la mayor cantidad de información de forma flexible sin preocuparse de la lógica para obtener los datos.

4.0.2 Indicadores y Cubos OLAP

El sistema cuenta con una función que genera los cubos OLAP automáticamente usando un esquema estrella. Los cubos son agregados dentro del mismo catalogo de Postgres pero en un esquema llamado 'cubos'. Esto permite que en el proceso de actualización de cubos, este esquema pueda ser borrado y creado nuevamente sin afectar el resto del sistema. A continuación se muestra como generar/actualizar los cubos usando la función interna del sistema desde la linea de comandos de Postgres:

```

minsal=# select * from cargar_cubos();
NOTICE: Inicio..
NOTICE: ALTER TABLE / ADD PRIMARY KEY will create implicit index
"cubos_indicador1_pk" for table "indicador1"
CONTEXT: SQL statement "ALTER TABLE cubos.indicador1 ADD CONSTRAINT
cubos_indicador1_pk PRIMARY KEY (id_fila)"
PL/pgSQL function crearcubos() line 39 at EXECUTE statement
NOTICE: Nueva Tabla cubos.indicador1
NOTICE: La columna id_area usara el catalago: ctl_area
NOTICE: La columna id_genero usara el catalago: ctl_sexo
NOTICE: La columna id_region usara el catalago: ctl_regiones
NOTICE: La columna id_municipio usara el catalago: ctl_municipio
NOTICE: La columna id_departamento usara el catalago: ctl_departamento
NOTICE: ALTER TABLE / ADD PRIMARY KEY will create implicit index
"cubos_indicador2_pk" for table "indicador2"
CONTEXT: SQL statement "ALTER TABLE cubos.indicador2 ADD CONSTRAINT
cubos_indicador2_pk PRIMARY KEY (id_fila)"
PL/pgSQL function crearcubos() line 39 at EXECUTE statement
NOTICE: Nueva Tabla cubos.indicador2
NOTICE: La columna id_area usara el catalago: ctl_area
NOTICE: La columna id_genero usara el catalago: ctl_sexo
NOTICE: La columna id_region usara el catalago: ctl_regiones
NOTICE: La columna id_municipio usara el catalago: ctl_municipio
NOTICE: La columna id_departamento usara el catalago: ctl_departamento
NOTICE: Terminado.
crearcubos
-----

(1 row)

```

4.0.3 Servidor de Gestión de Cubos OLAP

Actualmente el sistema utiliza un servidor experimental de cubos escrito en Python. Este servidor contiene tres elementos:

- 1- Un gestor de persistencia (SQLAlchemy). Este se conecta a nuestra base de datos y la descripción de los datos o modelo están contenidos en un arreglo JSON (archivo_modelo.json).
- 2- Un servidor Web para procesar peticiones REST. Este componente permite hacer consultas al cubo usando un URL desde AJAX. La instalación oficial utiliza Werkzeug, pero es posible anadirlo como un virtual host en apache.

3- Una utilidad de configuración y ejecución. SLICER es un componente que permite manipular el servidor desde la línea de comandos y su configuración se realiza en el fichero `slicer.ini`

A continuación se describen los pasos para instalar e integrar este servidor.

4.0.4 Instalar paquete Cubes de Python

En Debian squeeze con repositorio 'testing':

```
~#apt-get install python2.7 python-sqlalchemy python-werkzeug python-sqlite python-psycopg2 python-pip
```

Finalmente se debe usar un gestor de paquetes de Python para instalar el paquete del servidor OLAP 'Cubes'. La siguiente línea utiliza el gestor 'pip'.

```
~# pip install cubes
```

4.0.5 Configuración de modelo

El modelo contiene un listado de los cubos/indicadores disponibles en el sistema. El siguiente modelo es un ejemplo simplificado del cubo/indicador2 con los siguientes datos:

- La llave primaria del cubo/indicador2 es la columna `id_fila`
- Este indicador tiene dos dimensiones 'departamento' y 'area'.
- La dimensión área está asociada por medio de la llave foránea `indicador2.id_area`
- La dimensión departamento está asociada por medio de la llave foránea `indicador2.id_departamento`
- Este indicador tiene además dos campos ('anio' y 'edad') que no dependen de ninguna dimensión.
- La dimensión 'departamento' posee los campos descripción y abreviatura
- La dimensión 'area' posee los campos descripción y abreviatura

Este fichero es utilizado por SQLAlchemy para manipular el cubo y describe sus campos y llaves foráneas.

```
{  "name" : "cubos",
    "locale": "en",
    "cubes": [
```

```

    {
      "name": "indicador2",
      "key": "id_fila",
      "measures": {
        "calculo": {"label": "Total"}
      },
      "details": [
        {
          "name": "anio",
          "label": "Anio"
        },
        {
          "name": "edad",
          "label": "Edad"
        }
      ],
      "dimensions": [
        "area", "departamento"
      ],

      "joins": [

<div class="code code">
<pre class="code">    {
      &quot;master&quot;;: &quot;indicador2.id_area&quot;;,
      &quot;detail&quot;;: &quot;ctl_area.id_area&quot;;
    },</pre>
</div>

    {
      "master": "indicador2.id_departamento",
      "detail": "ctl_departamento.id_departamento"
    }
  ],
  "mappings": {
    "ft_indicador2.id": "ft_indicador2.id_fila"},
    "fact": "indicador2"
  }
],

  "dimensions": {
    "area" : {

```



```

        "name": "Area",
        "label": "Area de Poblacion",
        "key": "id_area",
        "levels" : [
            {
                "name": "area",
                "label_attribute": "descripcion",
                "attributes": [
                    {"name": "descripcion", "label": "area"},
                    {"name": "inicial", "label": "abreviatura"}
                ]
            }
        ]
    },

    "departamento" : {
        "name": "Departamento",
        "key": "id_departamento",
        "label": "Departamento / Provincia",
        "levels" : [
            {
                "name": "departamento",
                "attributes": [
                    {"name": "descripcion", "label":
"departamento"},
                    {"name": "abreviatura", "label":
"abreviatura"}
                ]
            }
        ]
    }
}

```

4.0.6 Configuración de la utilidad 'Slicer'

El iniciar el servidor se debe utilizar el siguiente comando:

```
~# Ruta_de_Instalacion/cubes/bin/slicer serve slicer.ini
```

El contenido del archivo slicer.ini es el siguiente:

```
[model]
path: Ruta_Modelo/mi_modelo.json

[server]
# Set writeable path for logging slicer info
log_level: info
prettyprint: true
backend: sql
reload: true

[workspace]
# SET DATABASE -> adapter://user:password@host:port/database
url: postgres://minsal:minsal@localhost:5432/minsal
schema: cubos
dimension_schema: public
dimension_prefix: ctl_
debug: true

# Denormalisation demo (see README.md)
#
denormalized_view_prefix = mft_
denormalized_view_schema = views
use_denormalization = yes
```

4.0.7 Consultando el Servidor OLAP

El servidor OLAP procesa consultas REST usando su API http, y responde con datos en formato JSON. El formato general a utilizar es el siguiente:

http://servidor:puerto/cubes/nombre_de_cubo/Consulta

Ejemplo:

```
~# curl "http://localhost:5000/cubes/indicador2/dimension/area"
{
  "depth": null,
  "data": [
    {
      "area.inicial": "U",
      "area.descripcion": "Urbano"
```

```

        },
        {
            "area.inicial": "R",
            "area.descripcion": "Rural"
        }
    ],
    "dimension": "area"

```

En este punto estamos listos para efectuar consultas desde el sistema de gestión de Indicadores usando AJAX. Un listado completo de las posibles consultas que se pueden hacer usando el servidor OLAP esta disponible en esta dirección:

<http://packages.python.org/cubes/server.html#http-api>

4.1 Funciones Auxiliares de PostgreSQL

Estas son funciones que utiliza el sistema, para poder crearlas es necesario instalar primero las funciones para manejar hstore y tablas pivot:

```
postgres=# create extension hstore;
```

```
postgres=# create extension tablefunc;
```

4.1.1 Crear/Actualizar Cubos

Para crear los cubos OLAP se debe llamar la funcion correspondiente:

```
postgres=# select * from cargar_cubos();
```

```
[....]
```

NOTICE: Terminado.

Es posible verificar las tablas/cubos generados en el paso anterior por medio de la siguiente consulta.

```
postgres=# SELECT table_name FROM information_schema.tables WHERE table_schema = 'cubos';
```

```
"" postgres
```

```
CREATE OR REPLACE FUNCTION cargar_cubos() RETURNS void AS $BODY$ DECLARE
indicador int; columnas text; cols text []; c text; col text; fk record; coltype record;
mycursor refcursor; mycursor2 refcursor; nombre_tabla text; myquery text; BEGIN SET
client_min_messages='INFO'; DROP SCHEMA IF EXISTS cubos CASCADE; CREATE
```

```

SCHEMA cubos; RAISE NOTICE 'Inicio.'; --Agrupar por indicador Para crear -- y poblar
tablas por indicador FOR indicador IN SELECT DISTINCT id_ficha_tecnica FROM fi-
cha_tecnica_variable_dato WHERE id_variable_dato IN (SELECT id FROM varia-
ble_dato WHERE id_origen_datos IN (SELECT id_origen_dato FROM fila_origen_dato
GROUP BY id_origen_dato)) LOOP

```

```

    --Obtener los campos comunes de este indicador
    FOR col IN SELECT skeys(datos) AS cosa FROM
fila_origen_dato WHERE
        id IN (SELECT min(id) AS cosa FROM fila_origen_dato
WHERE
        id_origen_dato=(SELECT min(id_variable_dato) as
tt
        FROM ficha_tecnica_variable_dato WHERE
id_ficha_tecnica=indicador))
    LOOP
        cols:= array_append(cols,col);
    END LOOP;

    --Crear lista de columnas
    columnas:=array_to_string(cols, ' text,') || ' text';
    columnas:='id_fila int, ' || columnas;

    --Crear y poblar tabla por indicador/ 'Fact Table' del cubo
    nombre_tabla:= 'cubos.indicador' || indicador;
    EXECUTE 'CREATE TABLE IF NOT EXISTS ' || nombre_tabla || '('
|| columnas || ')';
    -- RAISE NOTICE 'Se creo tabla % con columnas:\n %',
nombre_tabla,columnas;

    EXECUTE 'INSERT INTO ' || nombre_tabla || ' SELECT * FROM
    crosstab(''SELECT id,(each(datos)).key AS columna,
    (each(datos)).value AS valor FROM fila_origen_dato where
id_origen_dato
    IN (SELECT min(id_variable_dato) AS tt FROM
ficha_tecnica_variable_dato
    WHERE id_ficha_tecnica=' || indicador || ') '') AS ct(' || columnas
|| ')';

    columnas := '';
    cols:=ARRAY[]::text[];

```

```

END LOOP;

```

```
--Agrupar por indicador para crear estructura de estrella FOR indicador IN SELECT DIS-
TINCT id_ficha_tecnica FROM ficha_tecnica_variable_dato WHERE id_variable_dato
IN (SELECT id FROM variable_dato WHERE id_origen_datos IN (SELECT
id_origen_dato FROM fila_origen_dato GROUP BY id_origen_dato)) LOOP
```

```
    --Obtener los campos comunes de este indicador
    FOR col IN SELECT skeys(datos) AS cosa FROM
fila_origen_dato WHERE
        id IN (SELECT min(id) AS cosa FROM fila_origen_dato
WHERE
        id_origen_dato=(SELECT min(id_variable_dato) as
tt
        FROM ficha_tecnica_variable_dato WHERE
id_ficha_tecnica=indicador))
    LOOP
        cols:= array_append(cols,col);
    END LOOP;
```

```
    --Crear lista de columnas
    columnas:=array_to_string(cols, ' text,') || ' text';
    columnas:='id_fila int, ' || columnas;
    nombre_tabla:= 'cubos.indicador' || indicador;
    RAISE NOTICE 'Tabla % con Columnas: %',
nombre_tabla,columnas;
```

```
EXECUTE 'ALTER TABLE ' || nombre_tabla ||' ADD CONSTRAINT ' ||
replace(nombre_tabla,'.','_') ||'_pk PRIMARY KEY (id_fila)';
EXECUTE 'ALTER TABLE ' || nombre_tabla ||' ALTER COLUMN
calculo
    TYPE numeric(10,2) USING calculo::numeric(10,2)';
    RAISE NOTICE 'Nueva Tabla %', nombre_tabla;
```

```
    -- Crear estrella/relacios con otras dimensiones
    FOREACH c IN ARRAY cols LOOP
        IF (c <>'calculo') THEN
            OPEN mycursor FOR EXECUTE 'SELECT
significado_campo.catalogo FROM
        public.significado_campo WHERE
significado_campo.codigo = ''' || c ||''' AND
        significado_campo.catalogo IS NOT NULL;' ;

            FETCH mycursor INTO fk;
            --myVar := rec
            IF fk.catalogo IS NULL THEN
                RAISE NOTICE 'No se encontro Catalogo para:
```

```

%', c;

                                ELSE

                                RAISE NOTICE 'La columna % usara el
catalogo: %', c,fk.catalogo;
                                OPEN mycursor2 FOR EXECUTE 'SELECT
pg_typeof('||c||')::text as tipo
                                FROM public.'||fk.catalogo||' LIMIT 1';

                                FETCH mycursor2 INTO coltype;
                                myquery:= 'ALTER TABLE '|| nombre_tabla ||'
ALTER COLUMN '|| c ||
                                ' TYPE '|| coltype.tipo ||' using
                                '||c||':'||coltype.tipo;
                                --RAISE NOTICE 'Codigo que modifica campo
columna: %', myquery;
                                EXECUTE myquery;

                                myquery='ALTER TABLE '|| nombre_tabla ||'
ADD CONSTRAINT '|| c ||'_fk
                                FOREIGN KEY ('||c||') REFERENCES
public.'||fk.catalogo||' ('||c||') ON
                                DELETE NO ACTION ON UPDATE NO ACTION NOT
DEFERRABLE';
                                --RAISE NOTICE 'Codigo de llave foranea: %',
myquery;
                                EXECUTE myquery;
                                CLOSE mycursor2;

                                END IF;
                                CLOSE mycursor;
                                END IF;

                                END LOOP;
                                columnas :='';
                                cols:=ARRAY[]::text[];
END LOOP;

```

```

RAISE NOTICE 'Terminado.'; SET client_min_messages='WARNING'; END; $BODY$
LANGUAGE plpgsql VOLATILE COST 100; ""

```

4.1.2 Crear/Actualizar Catalogo Tiempo

```
postgres=# select * from crear_ctl_tiempo(2008,5);
```

NOTICE: Se creo tabla Tiempo de 5 anios a partir de 2008

```

```postgres CREATE OR REPLACE FUNCTION crear_ctl_tiempo(inicio integer DE-
FAULT 2006, anios integer DEFAULT 8) RETURNS VOID AS $$ DECLARE dias integer;
myquery text; BEGIN dias:=365*$2;

```

```

DROP TABLE IF EXISTS ctl_tiempo;

```

```

myquery:='CREATE TABLE ctl_tiempo AS SELECT * from (SELECT datum AS fecha,
extract(year FROM datum)::int AS Anio, extract(month FROM datum)::int AS Mes,
to_char(datum, "TMMonth")::character(12) AS NombreMes, extract(day FROM da-
tum)::int AS Dia, extract(doy FROM datum)::int AS DiaAnio, to_char(datum, "TM-
Day")::character(12) AS NombreDiaSemana, extract(week FROM datum)::int AS Sema-
naCalendario, to_char(datum, "dd. mm. yyyy")::character(12) AS FechaCorriente, "T" ||
to_char(datum, "Q")::int AS Trimestre, to_char(datum, "yyyy/Q")::character(6) AS Tri-
mestreAnio, to_char(datum, "yyyy/mm")::character(12) AS MesAnio, -- ISO calendar
year and week to_char(datum, "iyyy/IW")::character(8) AS SemanaAnioCalendario, --
Weekend CASE WHEN extract(isodow FROM datum) IN (6, 7) THEN "FinDeSemana"
ELSE "DiaDeSemana" END AS FinDeSemana, -- Feriados para El Salvador CASE WHEN
to_char(datum, "MMDD") IN ("0801", "0802", "0803", "084") THEN "Feriado" ELSE "Dia
Laboral" END AS FeriadoElSalvador, -- Periodos festivos del calendario CASE WHEN
to_char(datum, "MMDD") BETWEEN "0701" AND "0831" THEN "Vacación de Verano"
WHEN to_char(datum, "MMDD") BETWEEN "1115" AND "1225" THEN "Temporada Na-
videña" WHEN to_char(datum, "MMDD") > "1223" OR to_char(datum, "MMDD") <=
"1231" THEN "Vacación Navideña" ELSE "Normal" END AS Periodo, -- Fecha de inicio
de fin de semana datum + (1 - extract(isodow FROM datum))::integer AS IncioSemana,
datum + (7 - extract(isodow FROM datum))::integer AS FinSemana, -- Fecha de inicio
de fin de Mes datum + (1 - extract(day FROM datum))::integer AS InicioMes, (datum +
(1 - extract(day FROM datum))::integer + "1 month"::interval)::date - "1 day"::interval AS
FinMes FROM (SELECT "||$1||-01-01"::DATE + sequence.day AS datum FROM genera-
te_series(0,||dias||) AS sequence(day) GROUP BY sequence.day) DQ ORDER BY 1) as foo';
EXECUTE myquery;

```

```

ALTER TABLE ctl_tiempo ADD PRIMARY KEY (fecha);

```

```

RAISE NOTICE 'Se creo tabla Tiempo de % anios a partir de %',anios,inicio;

```

```

END; $$ LANGUAGE plpgsql; ```

```

Esta página se ha dejado vacía a propósito



# Lista de figuras

- 3.1 Esquema de la aplicación ..... 17
- 3.2 Diagrama ER1 ..... 18
- 3.3 Diagrama ER2 ..... 18