Documentación Base de datos

David Gómez Sánchez y Pablo Vicente Munuera

Abstract

Este documento contiene lo necesario para la creación de una base de datos relacional en PostgreSQL en la que se almacenarán datos provenientes de PFAM, JGI y HMMER3 para la posterior obtención de información concerniente a proteínas del organismo *Pseudovirgaria hyperparasitica*.

1 Getting started

Aunque está explicado como preparar el entorno para ejecutar nuestros programas, procederemos a hacerlo, igualmente, aquí.

1.1 Inserción de las tablas en la base de datos y ejecución del programa principal (main.py):

Lo primero que debe hacer es ejecutar, en una terminal, el script para acceder a la base de datos y crear las tablas necesarias para la ejecución de los programas:

chmod a+x init.sh

./init.sh

NOTA: los valores por defecto para psql son: localhost:5432:masterdb:masteruser:masterpass, si desea cambiarlos, por favor modifique el archivo init.sh

Realizado el primer paso, se deberá ejecutar el programa, escribiendo lo siguiente en una terminal:

python3.4 /main.py

Una vez dentro, observaremos un menú en pantalla que nos da la opción de escoger entre diferentes opciones. El usuario deberá seleccionar las tres primeras en orden, para despues poder optar a seleccionar las siguientes. Esto se debe a que las tres primeras opciones son programas de tipo analizador sintáctico de patrones (parser) capaces de obtener datos desde JGI(opción 1), PFAM(opción 2) y HMMER3 (opción 3) para acto seguido insertarlos en las tablas creadas en nuestra base de datos, mientras que los programas siguientes sirven para obtener información de dichas tablas una vez realizados los pasos 1, 2 y 3.

NOTA: se requerirá de un parámetro en las opciones 1,2 y 3 para los cuales el usuario puede indicar la ruta de los ficheros a introducir, o teclear 'intro', lo que introduce la ruta por defecto:

- *Datasets/Psehy1_GeneCatalog_proteins_20140829.aa.fasta* (Opción 1)

- *Datasets/Pfam-A.seed* (Opción 2)
- *Datasets/Pfam-A.hmm* (Opción 3)

1.2 Programas de consulta a la base de datos:

En este apartado se describen los programas que realizan consultas SQL a la base de datos (opciones 4-6 resultantes de la ejecución del programa principal **main.py**

- Opción 4: Programa que muestra por pantalla los dominios de las proteínas analizadas. Necesita como parámetro una proteína del organismo *Pseudovirgaria hyperparasitica* (ID o descripción de JGI) tecleada directamente en la terminal.
- Opción 5: Programa que muestra por pantalla los identificadores de InterPro de las proteínas analizadas. Necesita como parámetro una proteína del organismo *Pseudovirgaria hyperparasitica* (ID o descripción de JGI) tecleada directamente en la terminal.
- Opción 6: Programa que muestra por pantalla, para las proteínas etiquetadas como **kinasas, lyasas, canales iónicos, receptores** o **transportadores**, el número medio, máximo, mínimo y desviación estándar del número de dominios de *Pseudovirgaria hyperparasitica* . En este caso no es necesario ningún parámetro en la ejecución del mismo.

2 Realización de la práctica

2.1 Herramientas utilizadas

- Sublime Text o Geany. David utilizó Geany, mientras que Pablo utilizó Sublime Text, por cuestión de gustos, cumpliendo ambos perfectamente su función.
- Texworks. Es el programa utilizado para editar los textos de latex. Tiene función de compilado y te muestra en una ventana aparte el pdf generado.
- Github. Hemos utilizado un control de versiones digno, ya que en la experiencia de Pablo, es infinitamente mejor trabajar con uno, que trabajar sin él. David nunca había utilizado uno, pero en seguida le ha pillado el tranquillo y está muy agusto trabajando de esta manera.
- Python3.4. Es el lenguaje y versión establecida por el profesorado, debíamos ajustarnos a él.
- Postgresql. Había la posibilidad de utilizar SQLite, pero preferimos utilizar PostGreSQL debido a que nos parecía mas engorroso utilizar un gestor de bases de datos embebido para una práctica como esta.

2.2 Organización del trabajo

Aunque lo ideal hubiera sido utilizar *pair-programming* para la realización de esta práctica, se ha optado por repartirnos el trabajo e intentar quedar lo máximo posible para que no hubiera dudas, ni problemas, de modo que no se retrasara la velocidad de trabajo. Básicamente hemos intentado hacer ambos miembros de la práctica, tanto algo de lo que sería más bases de datos (consultas, diseño, etc...), como algo de la parte de programación, aprendiendo así las partes necesarias que se requerían para esta asignatura.

2.3 Diseño de base de datos

El diseño de base de datos nos ha llevado de cabeza durante la mayor parte del desarrollo del trabajo, aunque eso trataremos más debidamente en la sección: problemas encontrados.

Ha habido dos tablas claras desde el primero momento: la tabla jgi y la tabla Pfam. La primera representa al punto 1.a "Procedente de jgi" y corresponde a las proteínas y sus correspondientes secuencias y datos. Tiene como clave primaria el id, lo cual parece lógico, ya que es el elemento más representativo de cada proteína. Aunque en un primer momento se puso también el organismo, posteriormente se vió que el id cambiaba con el organismo, con lo que no era necesaria la dupla id-organismo como clave primaria. Al no tener algo en lo que basarnos para decidir los tipos, se ha tomado algo que no consumiera demasiado, pero que fuera más que suficiente. Por ello, en cosas que pueden ser muy largas, como la secuencia y la descripción, se han puesto *Text* como tipo de dato. Los *NOT NULL*, aún a sabiendas de que todo se debía meter, hay cosas que creemos que aunque no se metieran, no tendrían un efecto adverso, a la hora de las búsquedas y filtrados.

Por otro lado, la tabla Pfam también quedó bastante resuelta desde un principio. A parte de los datos que se nos informaba que debía contener esta tabla, lo único que había que pensar era la clave primaria y la posibilidad de claves alternativas. Las dos únicas cosas que no se pueden repetir es el ID y el accesion number. La combinación de ambas no era posible, ya que no para un mismo ID no va a haber 2 accesion number distintos, será siempre el mismo, con lo que se opta por poner una clave primaria (ID, pero podría haber sido perfectamente el accesion number) y una clave alternativa que será el accesion number. Los tipos de esta tabla, excepto el ID, que debió adecuarse a otra tabla, estaban específicados en la página web oficial de Pfam, así que no tuvimos que realizar el esfuerzo nosotros. Un dato curioso: intentando imitar lo que se dió en clase, se hizo una tabla para los accesion number y los posibles accesion numbers antiguos. Esta se acabó por eliminar, ya que no tenía mucho sentido en el entorno de la práctica.

El problema y la parte de las tablas que más cambios han sufrido han sido las correspondiente al punto 1.c "Datos provenientes de hmmer". Ha llegado a haber hasta 3 tablas, sólo para esta sección. Finalmente, se vió que con solo 2 tablas era suficiente. En primer lugar, tenemos la tabla hmmer, en la cual, guardamos la información de: el ID de la proteína con la que hacemos la query a hmmscan, que será un *INT* ya que es un número; la descripción de esta, que no tendrá tope de caracteres; y el e-value que

debido a que puede ser muy pequeño, lo pondremos como *float*. El ID de esta tabla será clave primaria y hará referencia al mismo ID de JGI. Por lo tanto, en la tabla HMMER deberán existir solo los ID, los cuales se encuentren también almacenados en la tabla JGI.

La tabla Domains, debería tener un apartado a parte en la sección de problemas. Solo esta tabla ha sufrido más cambios que todas las demás juntas. Esto es debido en gran parte a que no teníamos la experiencia necesaria, y hasta que no supimos como afrotar y que debiamos hacer en el punto 2.3 (e incluso en mitad del desarrollo) no supimos el correcto diseño de dicha tabla. En un principio iban a ser dos tablas, pero al final, quedó reducida a un única tabla. En ella tendriamos, el ID del dominio, que será, básicamente, un autogenerado que en postgreSQL se denomina *serial*, el cual será clave primaria. Como datos *INT* tenemos: startAA, que corresponde al número dónde empieza el aminoácido alineado del dominio; endAA, es la posición dónde acaba el alineamiento; y el score, que es la puntuación obtenida en el alineamiento (como es de bueno o malo). También tenemos como elemento *float*, el evalueInd, que corresponde al independent e-value. Estos datos eran lo que se pedía en la segunda parte del punto 1.c. Para que se relacionasen estos datos tanto con la tabla Hmmer, como con los datos en Pfam, debemos añadir tres campos más, correspodientes a: La proteína query, representada por IDQuery que hace referencia a la columna ID de Hmmer, la cual, a su vez, llevará a la proteína de JGI; Los dominios target, que referencian a la tabla Pfam, y, por lo tanto, tendremos que tener sus correspondientes valores, AccTarget (accesion number del dominio) y NameTarget (ID de Pfam). Ambas relaciones serán de 1 a 1 desde la tabla Domains hacia las otras tablas.

2.4 Problemas encontrados

2.4.1 Problemas con python y SQL

En mi caso (David, perfil bio) he tenido ciertos problemas a la hora de enfrentarme a la sintaxis del código en python y saber resolver programas de parseo, por simple deficiencia en conocimientos de programación, que finalmente han sido subsanados en su gran mayoría por el compañero Pablo. Por otra parte, ciertas sentencias de SQL me han llevado algo de tiempo por el mismo motivo; sin embargo me he apoyado en Pablo, en el manual de PostgreSQL y en algún foro del tipo stackoverflow para sacarle mas partido y ampliar conocimientos, con lo que siento que he aprendido bastante.

2.4.2 Problemas con el diseño de las tablas

Como ya se ha mencionado anteriormente, la parte de las tablas correspondientes a la información sacada desde *hmmscan*, ha sido siempre la más compleja y cambiante a lo largo del proyecto. En un primer momento se pensó que debía haber tres trablas: una para hmmer (eso no ha cambiado), y dos para los dominios. Esta se completaba con la información básica de un dominio en una tabla denominada dominio y otra tabla con las relaciones oportunas con pfam y hmmer (que sería la tabla domains). Pero, esto tenía

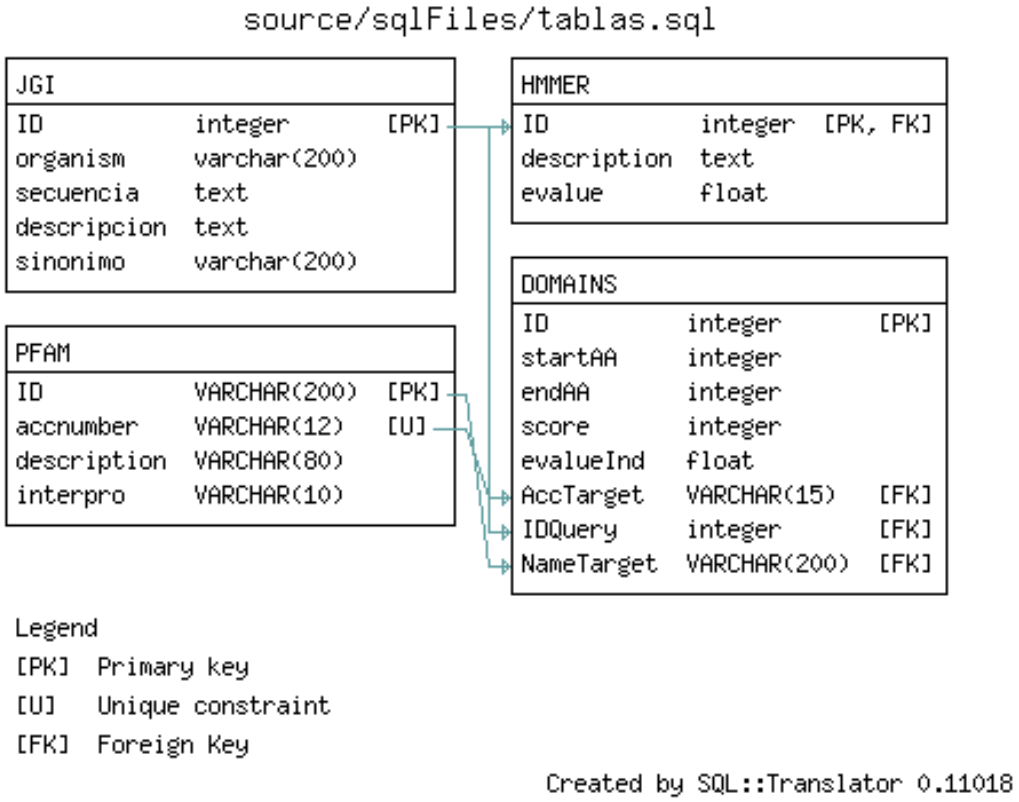

Figure 1: Diseño final de la base de datos

una pega, no merecía la pena tener dos tablas, cuando en una sola, podrías tener las dos y sin apenas redundancia, con lo que se decidió ponerla en una sola. A parte de los múltiples cambios que han sufrido esta tablas, lo demás no ha tenido mayor complejidad.

3 Conclusión

Como conclusión podemos decir que la práctica ha ayudado a la parte biológica a aumentar sus habilidades programáticas y afrontar problemas que antes no se les habían planteado. Para la parte informática, ha sido útil para refrescar conceptos olvidados y para ver como se aplican conceptos informáticos en temas biológicos.