MANUAL DE USUARIO

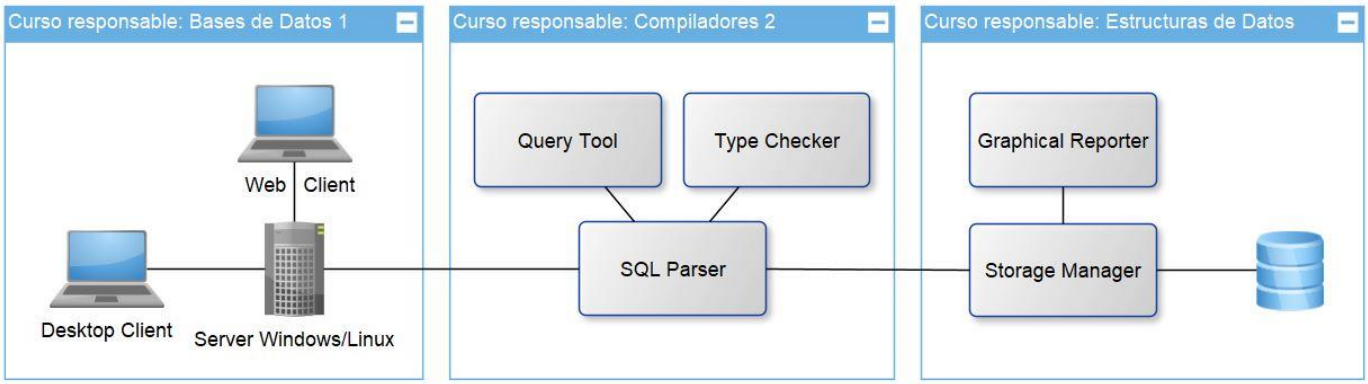
TytusDB-SQL Parser

Que es TytusDB yu

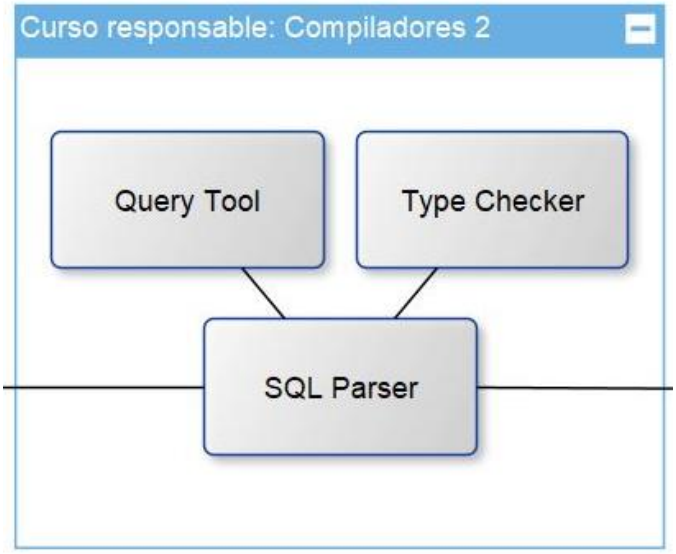
Es un proyecto Open Source para desarrollar un administrador de bases de datos. Está compuesto por tres componentes interrelacionados: el administrador de almacenamiento de la base de datos, que estará a cargo del curso de Estructuras de Datos; el administrador de la base de datos, que estará a cargo del curso de Sistemas de Bases de Datos 1, este administrador se compone a su vez de un servidor y de un cliente; y el SQL Parser, que estará a cargo del curso de Organización de Lenguajes y Compiladores 2.

TytusDB – SQl Parser

La siguiente figura muestra la integración entre los tres cursos desarrollando Tytus. Este manual está dedicado solo al TytusDB-SQL Parser



TytusDB-SQL Parser tiene la siguiente estrutura:



- Es un intérprete para el subconjunto del lenguaje SQL mediante la traducción dirigida por la sintaxis.

- Herramienta utilizada PLy de Phyton para la traducción.

Lenguaje de programación

El lenguaje en el cual se trabajó todo el proyecto es Phyton, utilizándose algunas bibliotecas adicionales como PLY, tabulate, entre otras, siempre verificando que la Licencia coincida a la Licencia general del proyecto que es MIT.

Licencias y convenio

El proyecto está diseñado por el catedrático bajo una licencia Open Source, específicamente MIT. Por convenio, los estudiantes aparecerán como contribuidores junto con el copyright. Además, cualquier biblioteca autorizada también se debe colocar la licencia y el copyright en el archivo LICENSE.md en su carpeta respectiva.

Este proyecto genera diferentes reportes:

- Reportes de errores léxico, sintácticos y semánticos.
- Reporte de tabla de símbolos.

- Reporte de AST.
- Reporte gramatical

Para que el proyecto pueda funcionar debe crearse un archivo de entrada en formato de texto para poder evaluar las instrucciones, estas instrucciones deben tener la sintaxis general de SQL.

Para crear este archivo de entrada se puede revisar la documentación PostgreSQL 13.1, para escribir las instrucciones. Este archivo de prueba puede ser manipulado las veces que sean necesarias previo al a ejecución del programa.

Ejemplo de un Archivo de Entrada:

select codigo,nombre,punteo from alumnos where punteo>40;

select * from tbcolaborador WHERE fecha = now() and codigo = 5;

CREATE TABLE tbempleadopuesto

(

idempleado integer not null,

idpuesto integer not null,

departamento area

);

```
CREATE TABLE tbempleadopuesto
(
        idempleado integer not null,
        idpuesto   integer not null,
        departamento area
);

 alter table tbempleadopuesto
 add constraint FK_empleado
 foreign key (idempleado)
 references tbempleado(idempleado);

 alter table tbempleadopuesto
 add constraint FK_empleado
 foreign key (idempleado)
 references tbempleado(idempleado);
```

```
insert into tbempleadopuesto values(1,1,'ADMINISTRACION');
insert into tbempleadopuesto values(2,1,'CONTABILIDAD');
insert into tbempleadopuesto values(3,3,'CONTABILIDAD');
insert into tbempleadopuesto values(4,6,'VENTAS');
insert into tbempleadopuesto values(5,6,'VENTAS');


UPDATE tbempleadopuesto SET idpuesto = 2 where idempleado = 2;
```

Al tener revisado y listo el archivo de prueba, se puede proceder a correr el programa desde el archivo “principal” que es el que se encarga de hacer llamar al archivo de prueba, el cual empieza a generar el AST y también de interpretar las instrucciones enviadas een el archivo.

La siguiente figura es un ejemplo de un AST pequeño generado de una instrucción SELECT.