Trabajo práctico 1:
"Base de Datos"

Normativa

Límite de entrega: Miercoles 29 de Agosto, 23:59hs. Enviar PDF al mail: `algo2.dc+TP1@gmail.com`
Normas de entrega: Ver "Información sobre la cursada" en el sitio Web de la materia.
(`http://www.dc.uba.ar/materias/aed2/2017/2c/informacion`)
Versión: 1.0a del 11 de agosto de 2017 (ver TP2_Changelog.txt)

Corregir mail de entrega

Corregir fecha de entrega

Enunciado

Una base de datos es un repositorio de información almacenada en una o más ***tablas***. Una tabla consiste en un conjunto de ***campos*** identificados por nombre. Los campos pueden ser de tipo *Nat o String*. Por ejemplo, en una hipotética base de datos universitaria, la tabla **Alumno** tiene los campos {Nombre: String, Edad: Nat, DNI: Nat} y la tabla **Inscripción** tiene los campos {Materia: String, DNI: Nat}.

En una tabla se pueden insertar **registros**. Un registro consiste en un conjunto de valores, uno para cada campo de la tabla. Siempre se respeta el tipo de datos asignado a cada campo. Por ejemplo, en **Alumno** podemos insertar los registros {"March", 30, 32658} y {"Gerva", 30, 12345}. El orden de inserción es irrelevante. Una tabla no puede tener registros duplicados (es decir, múltiples registros con todos sus campos iguales).

Toda tabla se crea designando uno o más campos como **claves**. En una tabla no puede haber dos registros que tengan los mismos valores para todos los campos **clave**. Si en la tabla **Alumno** quisieramos identificar a los alumnos por libreta universitaria, la tabla debería tener los campos {Nombre: String, Edad: Nat, NroLU: Nat, AñoLU: Nat} con NroLU y AñoLU como campos claves. De esta forma, la tabla podría tener los registros {"March", 32, 65, 16}, {"Gerva", 24, 122, 11}, {"March", 38, 65, 3}. No obstante, no podría agregarle el registro {"Analía", 19, 122, 11}.

Objetivo

El objetivo del trabajo práctico es implementar las operaciones de una **base de datos**. Para ello se espera que desarrollen las clases de C++ necesarias.

En una base de datos se debe poder agregar en cualquier momento una tabla nueva con un nombre que la identifique. A su vez, se le debe poder preguntar si dado una tabla de la base de datos y un registro, el último puede ser insertado en la primera; teniendo en cuenta las restricciones especificadas para agregar registros a una Tabla. La base de datos también se debe permitir agregar un registro a una tabla dada (si se cumplen las condiciones anteriores). Finalmente, una base de datos debe permitir realizar una **búsqueda** en una tabla.

En una base de datos una **búsqueda** devuelve una copia de la tabla sobre la que se busca con un subconjunto de los registros originales que cumplen el **criterio** de búsqueda. Un criterio se define como un conjunto de restricciones sobre los campos de la tabla. Una restricción se define con un campo, un valor y una opción para definir si filtrar los registros para los que en ese campo se encuentra ese valor o los registros para los que en ese campo no se encuentra ese valor. De esta forma, una base de datos debe poder decirnos si un criterio es válido para una tabla dada y luego poder ejecutarlo. Un criterio será válido si sus campos son campos de la tabla y si los valores de los mismos son del mismo tipo que el campo de la tabla. Finalmente, nos interesa que una base de datos nos diga cuál es el criterio de búsqueda que más se utilizó.

Requisitos y consideraciones

- Para la implementación de la base de datos y el resto de las clases necesarias, se espera que codifiquen (o codeen) clases de C++.
- Se espera que puedan interpretar cada clase implementada como un TAD. Por tanto, deben disponer para cada una de las funciones públicas de la clase si estas son `generadores`, `observadores`, u `otras operaciones`. Esto puede realizarse mediante comentarios en la función en el `.h` correspondiente.
- Además, todos las clases implementadas deberán tener una implementación propia del operator==.
- Se considerará como parte de la evaluación la prolijidad del código implementado. Esto incluye, pero no se limita a:
  - Modularización de las funcionalidad en clases
  - Modularización de las funcionalidad en funciones
  - Formato del código
  - Claridad de las ideas
- Deberán estar implementadas todas las funcionalidades de la base de datos descripta en la sección Objetivo.

Como parte del enunciado la cátedra provee la implementación de las clases Tabla, Registro y Dato. La implementación a realizar como solución del práctico deberá agregar código a proyecto de C++ provisto por la cátedra. Además, cada clase implementada cuanta con su especificación en TAD que también se encuenta disponible junto con el proyecto.