Trabajo Práctico 2:
Análisis de distribuciones de datos

Base de Datos - DC - FCEN - UBA

www.dc.uba.ar/materias/bd/

1 Introducción

La habilidad para elegir un mejor método para resolver una consulta es una característica fundamental en los motores de bases de datos. La diferencia en tiempo puede convertir una operación prohibitiva en una realizable. Es por eso que los motores de bases de datos estudian desde distintos ángulos cómo optimizar cualquier consulta posible.

Uno de los muchos enfoques consiste en entender *cómo* son los datos almacenados. Esta información resulta valiosa pues permite tomar decisiones sobre qué índices usar, teniendo un impacto contundente en la performance. El siguiente caso ejemplifica el valor de lo antes dicho.

Una base de datos almacena, entre otras cosas, un padrón electoral. En esta, cada ciudadano tiene diversos campos que lo caracterizan (nombre, dni, lugar de residencia, etc). A su vez, se cuenta con 2 índices distintos: uno sobre el año del nacimiento y otro sobre el distrito donde vive el ciudadano. Se quiere resolver la siguiente query:

Listar todos los ciudadanos que hayan nacido en 1985 y vivan en el distrito nacional 19.

Para resolverla, podrían armarse los dos siguientes métodos de evaluación (entre otros):

- Usar el índice sobre año de nacimiento y por cada uno de los que coincidan con 1985, filtrar los que pertenezcan también al distrito nacional 19.
- Usar el índice sobre el distrito nacional 19 y para cada uno, filtrar los que hayan nacido en 1985.

Para decidir cuál de los dos índices usar, necesitaríamos estimar cuántos registros cumplen cada condición por separado: año de nacimiento igual a 1985 y cuántos pertenecen al distrito nacional 19. Teniendo esta información podríamos tomar la mejor decisión. Una opción sería, recorrer toda la tabla contando cuántos cumplen la primer condición y, luego, la segunda. Esto sería muy lento, yendo explícitamente en contra de la motivación de estudio. No conociendo los datos, solo podríamos hacer la siguiente estimación:

- Un ciudadano puede tener entre 16 y 110 años (a lo sumo). Por lo tanto, la probabilidad de que alguien nazca en un año particular es $\frac{1}{100-16} \approx 0.0119$
- Distritos nacionales hay 40, tomando el mismo razonamiento, la probabilidad de que alguien pertenezca a uno en particular es $\frac{1}{40} \approx 0.0250$

En consecuencia a esta estimación deberíamos elegir el uso del índice sobre el año de nacimiento, que nos dejería solo el 1.1% de los registros e iterar sobre estos (en el otro caso, nos quedaríamos con el 2.5% y deberíamos recorrer más). El razonamiento anterior funcionaría si se cumpliese la

(implícita) hipótesis que usamos: los datos están **distribuidos uniformemente** (hecho trivialmente falso tratando de distribuciones sobre edaded y poblaciones).

El ejemplo anterior ilustra la necesidad de conocer los datos que manejamos y cómo esto permite tomar buenas (o malas) decisiones que impactan en la performance. Es por esto que este trabajo práctico tiene como objetivo estudiar métodos para entender las **distribuciones** subyacentes a los datos para la correcta toma de decisiones.

2 Trabajos relacionados

En función del objetivo propuesto, nos centraremos en algunas técnicas de la literatura presentados en los siguientes dos trabajos:

- G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. *Proc. of ACM SIGMOD Conf.* pages 256-276, 1984.
- H. To, C. Kuorong, C. Shahabi. Entropy-based histograms for selectivity estimation. *Proc. of the 22nd ACM international conference on Conference on information & knowledge management.* pages 1939–1948. 2013.

3 Estimadores

Como se ejemplificó en la introducción, conocer los datos alamacenados repercute fuertemente en la toma de decisiones y, por ende, en la performance. Es por eso que tendrán que implementar estimadores. Los estimadores son clases que permiten, a partir de la creación de una estructura, resolver diversas consultas que estimen cuántas tuplas (o valores de la columna/vector en este caso) satisfacen una condición. Usando el ejemplo de la introducción desearíamos realizar dos consultas (a dos estimadores distintos, pues predican sobre los años (que es una columna) y sobre el distrito (otra colunma)). La primera sobre el año de nacimiento por igualdad con 1985 y otra por igualdad (para el segundo estimador) con distrito 19.

Es importante entender que la construcción del estimador puede ser costosa pero una vez que esté creado las consultas (estimaciones) serán rápidas y muchas. Es por esto que vale la pena contar con una estructura costosa con consultas veloces.

La figura 1 resume el ciclo de vida de los estimadores.

1. **Etapa de Construcción**: Para poder realizar las consultas es necesario primero armar una estructura que de soporte a éstas. Dicha estructura debe almacenar información que permita tomar decisiones. La función encargada de crearla tendrá como input una tabla, una columna y un parámetro de estimación. Por ejemplo, para el caso del histograma clásico descripto en el trabajo de Piatetsky-Shapiro los inputs serán: la tabla, una columna y un parámetro llamado $PARAM$ que representa la cantidad de *bins* que tendrá el histograma resultante.

   Existe una **fuerte restricción** : No pueden suponer que la columna entera entre en memoria. Es decir, para la construcción de la estructura deben basarse fuertemente en los servicios del motor y usar todas las herramientas que éste provee (contar tuplas que satisfagan condiciones, encontrar mínimos, máximos, etc.) y no resolver estos problemas usando Python.

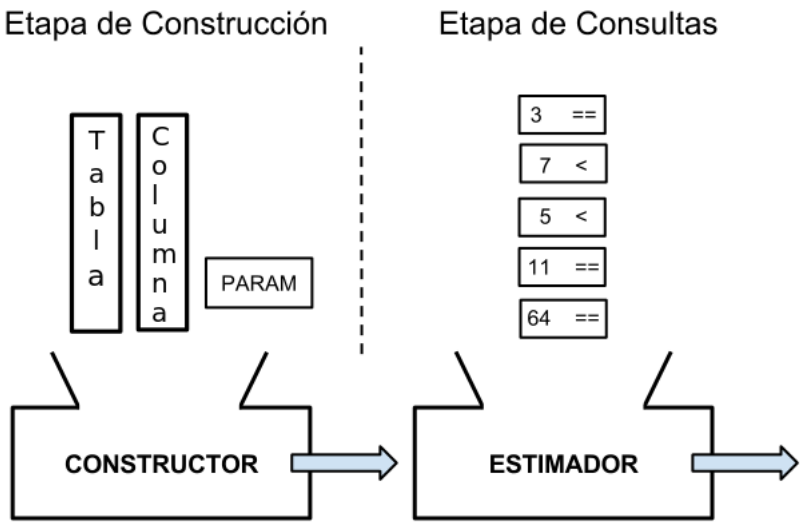

Fig. 1. Ciclo de vida de los estimadores.

2. **Etapa de consulta**: Una vez creada la instancia del estimador, éste deberá resolver los dos tipos posibles de consultas: comparación por igual y comparación por mayor. Para resolverlas deberán basarse solo en la estructura creada anteriormente. Esta estapa no tiene restricciones.

   El output será la *selectividad* estimada. Donde *selectividad* dada una condición representa la fracción de tuplas que satisfacen la condición (ver Piatetsky-Shapiro)

4 Ejercicios

1. **Conexión a la BD**

   El motor de base de datos que vamos a usar es SQLite[1]. Usaremos este *motor* por dos características: es accesible desde muchos lenguajes de programación, en particualar Python [2], y a su vez simplifica la conexión al motor pues guarda la BD en un archivo.

   A modo de práctica, deberán crear el modulo *interfaz_bd.py* y resolver las consultas que se detallan:

   (a) Crear una BD.
   (b) Crear una tabla en la BD anterior e insertar registros.
   (c) Realizar distintas consultas sobre la tabla creada y mostrar los resultados.
   (d) Realizar actualizaciones sobre los datos cargados.
   (e) Borrar algunos de los registros cargados.
   (f) Mostrar todas las tablas definidas en la BD.

[1] http://sqlite.org/
[2] http://docs.python.org/2/library/sqlite3.html

2. **Estimadores**

Deberán implementar **tres** tipos distintos de estimadores. Los estimadores son clases que deben cumplir con tres métodos: un constructor de la estructura que se usará con cada consulta que se le haga, un estimador por igualdad y un estimador por mayor. Más detalle en el apéndice (6.1).

(a) Complete la clase *ClassicHistogram* del archivo *estimators.py* . Dicha clase implementa el método de estimación basado en histogramas clásicos (en Piatetsky et. al. 4.1).

(b) Complete la clase *DistributionSteps* del archivo *estimators.py* . Dicha clase implementa el método de estimación basado en *Distribution Steps* (en Piatetsky et. al. 4.2).

(c) Complete la clase *EstimatorGrupo* del archivo *estimators.py* . Dicha clase implementa el método de estimación basado en *su creatividad*, para esto puede inspirarse en el trabajo *Entropy-based histograms for selectivity estimation*(en H. To et. al.) si lo prefieren.

3. **Análisis de métodos**

(a) Preguntas introductorias

i. Dar 2 ejemplos de la vida cotidiana de distribuciones normales.

ii. Dar 2 ejemplos de la vida cotidiana de distribuciones uniformes.

iii. Conseguir 2 datasets reales que ejemplifiquen una distribucion normal y otra uniforme, explicar y graficar sus histogramas.

(b) Estimadores

Podemos definir la *performance* de un estimador, para un dataset particular, como la diferencia absoluta entre la selectividad estimada y la selectividad real. Una buena manera de resumir esta información es tomando la media de las diferencias absolutas y el error relativo.

Para los siguientes ejercicios, construyan datos de test con el fin de justificar sus respuestas, grafiquen para acompañar sus argumentaciones, creen lo necesario para armar experimentos que les permitan contestar las preguntas.

i. Dados los métodos “Classic Histograms” y “Distribution Steps”. Cuál esperaría que tenga mejor performance en la estimación para un parámetro fijo (en caso del primer método, el parámetro es la cantidad de *bins* y, en el segundo, la cantidad de *steps*? Justificar.

ii. Analice cómo impacta la variación de los parámetros de los estimadores (*bins* para el caso de “Classic Histograms”, *steps* para “Distribution Steps” y los que usen en su estimador) en distintas distribuciones. Cuál es la ventaja y desventaja de cambiar estos valores? En la única variable donde impactan estos cambios es en la performance? Qué pasa con el tiempo que cuesta constuir la estructura o la memoria que se usa para guardarla? Justifique (Sugerencia: El grafico *error bar* puede ser útil como herramienta.)

iii. Analice cómo impacta el uso de distintas distribuciones como entrada para los tres estimadores. Justifique.

(c) Datasets de prueba

Para este punto la cátedra proverá una BD (en un archivo de SQLite3, *db.sqlite3*) con datos de prueba. La misma cuenta con una sola tabla que contiene 10 columnas de Integers, llamadas "c0", "c1", ..., "c9". Cada una de ellas cuenta con datos provinientes de una distribución particular.

i. Para cuáles columnas de la BD de prueba la diferencia de performance entre los distintos estimadores es significativa? Para determinar esto use *Student Test* [3] apareado.

ii. En *Distribution Steps* los autores proponen una cota de error para la selectividad estimada. Muestre como esto se cumple para cada columna de la BD de prueba. Grafique y analice como varía la pérformance para cada columna en función del parámetro de *Steps*.

Consideraciones:

Todos los gráficos que presenten deben ser **reproducibles**. Por lo tanto, deben crear un *script* que los genere de manera automática y que permita cargar otros datos, distintos a los que usaron para generarlos. Sugerimos usar PyLab[4] para graficar.

5 Normas de Aprobación y Condiciones de Entrega

El trabajo práctico deberá ser resuelto en grupo (el mismo que para el trabajo práctico 1). La entrega del trabajo consiste en el código más un informe. Podrán desaprobar ambas partes por separado, sin embargo, no se aceptará para ninguna entrega una parte sola. Es decir, tienen que entregar informe para que se les corrija el código y viceversa. El informe debe incluir, como mínimo, las siguientes secciones:

- descripción detallada de toda la información que consideren necesaria para entender su estimador,
- análisis intuitivo del tipo de distribuciones con los que intuyen que su estimador se comportará de mejor manera,
- información y métodos utilizados para responder los ejercicios,
- comparación exhaustiva con los otros estimadores; presentar esta información en forma gráfica y escrita,
- sección de discusión.

Tendrán el mismo tutor asignado que en el trabajo práctico 1, se recomienda consultar fuertemente con su tutor. Sin embargo, pueden realizar preguntas generales a cualquier docente.

La fecha límite para la primera entrega es el **Miercoles 11 de Junio**, la fecha límite para la segunda entrega es el **Viernes 11 de Julio** (se recomienda fuertemente entregar antes de esta fecha). Adicionalmente, habrá una fecha optativa de chequeo del estimador propio con el tutor asignado el **Miércoles 28 de Mayo**; el objetivo de esta etapa es validar el estimador propuesto antes de

[3] http://en.wikipedia.org/wiki/Student's_t-test
[4] http://wiki.scipy.org/PyLab

avanzar con el desarrollo del TP (si bien esta etapa es optativa, la cátedra recomienda fuertemente utilizarla).

Para los que quieran podrán participar en una **competencia** con su tercera implementación (o una mejora si quieren cambiarla luego de entregar). En la competencia competirán todas las implementaciones y los grupos con las mejores estimaciones tendrán premios.

Resumen:

- Fecha de Presentación: Viernes 16 de Mayo
- Fecha de Entrega: Miercoles 11 de Junio
- Fecha de Última de Entrega de Recuperatorio: Viernes 11 de Julio

6 Apendice

6.1 Módulos

estimators.py **:** Este módulo es el encargado de resolver las consultas de *selectividad*. Se implemetan tres clases (dos de la literatura más una inventada por ustedes). Se deben respetar tres funciones por cada clase, (pudiendo agregar las que les hagan falta) estas son:

`def __init__(self, bd_name, table_name, column_name, parameter=10)`

Esta función es la que inicializa las nuevas instancias de la clase. Toma la información de dónde sacar los datos (base de datos, tabla y columna) y *parameter* representa el parámetro que usa cada método de estimación (cantidad de *bins*, para el *Classic Histograms*, cantidad de *step*, para el método de *Distribution Steps*).

`def estimate_equal(self,value)`

Esta función devuelve la *selectividad estimada* para la condición por igualdad al parámetro *value*.

`def estimate_greater(self,value)`

Esta función devuelve la *selectividad estimada* para la condición por mayor al parámetro *value*.

Ejemplo de uso:

```
# Creo un estimador del tipo Classic Histogram conectandome a la base
# de datos bd1, a la tabla table1 a la columna c2 usando 13 como parametro
aClassicEst= estimators.ClassicHistogram('db1','table1','c2',13)

# Obtengo la selectividad para la operacion igualdad  con valor a 5
selectividad = aClassicEst.estimate_equal(5)
```

6.2 Librerias recomendadas

A continuación se listan paquetes que pueden serles útiles para la resolución del trabajo práctico.

- numpy (aka np): http://www.numpy.org/
- pylab: Ejemplos de cómo plotear, http://matplotlib.org/examples/index.html
- scipy: Test de hipótesis, http://docs.scipy.org/doc/scipy/reference/stats.html
- sqlite3: Cómo conectarse, ejecutar queries, etc http://docs.python.org/2/library/sqlite3.html