



**UNIVERSIDAD DE BUENOS AIRES**

**Departamento de Computación**

**Facultad de Ciencias Exactas y Naturales**

## **Bases de Datos**

### **Trabajo Práctico 2**

**10 de noviembre de 2015**

<b>Integrante</b>	<b>LU</b>	<b>Correo electrónico</b>
Maurizio, Miguel Sebastián	635/11	miguelmaurizio.92@gmail.com
Prillo, Sebastián	616/11	sebastianprillo@gmail.com
Tagliavini Ponce, Guido	783/11	guido.tag@gmail.com

Índice

1. Introducción	1
2. Ejercicio 1	2
3. Ejercicio 2	2
4. Ejercicio 3	2
4.1. Escenarios adecuados para sharding . . . . .	2
5. Conclusión	2

# 1. Introducción

En este trabajo práctico diseñamos e implementamos una base de datos para el Registro Único de Accidentes de Tránsito (RUAT), sistema que está que está preparando el Gobierno Nacional, para registrar y analizar información sobre accidentes e infracciones de tránsito ocurridos en el país.

Por un lado, el sistema registra todos los datos relacionados con siniestros de tránsito, lo cual abarca:

- vehículos involucrados;
- conductores involucrados;
- testigos;
- localización;
- modalidad del siniestro (atropello, vuelco, etc.);
- tipo de colisión;
- denuncia radicada por el hecho;
- estudios y peritajes hechos.

Por otro lado, registra infracciones de tránsito, y más específicamente:

- vehículo involucrado;
- conductor involucrado;
- localización;
- tipo de infracción;

Además, el sistema registra datos sobre los vehículos, personas y las vías nacionales. Sobre los vehículos, almacena:

- categoría de coche (gama media, gama alta, etc.);
- tipo de vehículo (auto, camión, moto, etc.);
- seguro automotor y su tipo;

Sobre las personas, almacena:

- datos personales;
- autos de los cuales es dueña;
- cédulas (verdes y azules) que posee;
- licencias de conducir que posee;
- antecedentes penales;

Finalmente, sobre las vías nacionales, el sistema almacena:

- tipo de vía según el tramo (calle, avenida, autopista, etc.);
- extensión del tramo;

La elicitación de estos requerimientos proviene, principalmente, de la lectura del enunciado del trabajo práctico, que contiene toda esta información.

En las subsiguientes secciones presentamos cada una de las etapas del diseño y la implementación: el DER, su transformación al MR, y la implementación de queries y triggers. Para la implementación, decidimos utilizar el motor *SQLite*.

## 2. Ejercicio 1

## 3. Ejercicio 2

## 4. Ejercicio 3

### 4.1. Escenarios adecuados para sharding

Para que un atributo sea un buen candidato para aplicar la técnica de sharding, debe ser tal que al particionar los datos por ese atributo, las clases que se formen sean de tamaños similares. Esto permite que la carga sobre los shards esté bien distribuída, asegurando una mejor performance de I/O. Por ejemplo, si los accesos se distribuyen uniformemente sobre los shards, entonces la latencia y el tiempo de respuesta de los mismos será baja, puesto que evitamos cuellos de botella.

Para ilustrar esto, pensemos en un mal ejemplo de sharding. Consideremos una base de datos que contenga los datos de los alumnos del Departamento de Computación de la facultad. Si hacemos sharding sobre el atributo *género*, tendremos dos shards completamente asimétricos en su tamaño. Claramente, la enorme mayoría de los accesos serán sobre el shard de alumnos de sexo masculino, lo cual tiene un obvio impacto en la performance.

Algunos ejemplos de un buen sharding son los siguientes:

1. Sharding por atributo *género* en una base de datos de la población de un país.
2. Sharding por atributo *país de residencia* en una base de datos de personas de una red social.
3. Sharding por atributo *año de nacimiento* en una base de datos de personas nacidas durante cierto período de tiempo, fijo. Por ejemplo, durante la dictadura de 1976.

Finalmente, vale la pena destacar que el sharding puede ser de utilidad no sólo para asegurar balanceo de carga. En ciertos sistemas, son habituales las consultas que necesitan revisar únicamente un subconjunto de todo el universo de elementos. En estos casos, es útil hacer sharding de modo tal de que uno de estos subconjuntos que nos interesan estén separados en shards. Retomando el ejemplo de una base de datos de

## 5. Conclusión

El ejercicio del diseño una base de datos de una escala considerable, nos dejó algunas enseñanzas. Por un lado, nos permitió adquirir mayor confianza en el trabajo con el modelado de problemas vía bases de datos relacionales, y la implementación en SQL. Pero más importante, nos permitió sentir en carne propia las dificultades del diseño, que hacen que de esto un proceso iterativo, en el cual sólo se llega a la solución final a través de refinamientos. En nuestro caso particular, la confección del DER la desarrollamos a lo largo de varias semanas,

realizando sucesivas veces el antedicho refinamiento. Gracias a esto, la implementación fue fluida, sin mayores inconvenientes, más allá de los del aprendizaje del lenguaje de consultas.