

**Universidad de Buenos Aires**

**Facultad de Ingeniería**

**Departamento de Informática**

***Organización de Datos (75.06)***

**Voto Electrónico**

Cuatrimestre y año: 2do Cuatrimestre 2011

Docente a cargo del TP:Nicolás Pablo Fernández Theillet

Grupo: Lamas

Fecha de Entrega:

Integrantes: 2011-10-22

|  |  |  |
| --- | --- | --- |
| ***Padrón*** | ***Nombre*** | ***Email*** |
| 91187 | Gonzalez Durand, Juan Manuel | jmanuel.gonzalez.durand@gmail.com |
| 90762 | Ostrowsky, Gabriel | gaby.osto@gmail.com |
| 90728 | Schenkelman, Damián | damian.schenkelman@gmail.com |
| 91045 | Torrado, Alejandro | aletorrado@gmail.com |
| 90884 | Zamudio, Gonzalo | ahogadosderazon@gmail.com |

Índice

[Diseño 3](#_Toc304593810)

[Distrito 3](#_Toc304593811)

# Diseño

A continuación se detallan las decisiones de diseño tomadas para la organización de los archivos de las diferentes entidades del problema. Para cada una de ellas se detalla:

* La organización que se utilizara para guardar los datos de relacionados con la misma.
* Las razones por la que se decidió dicha organización.
* Características particulares dentro de la organización elegida (ej: función de dispersión para el Hash).
* Estructura de los registros utilizados (tanto en índices como para guardar datos).
* Estructura de los bloques utilizados.
* Una imagen o gráfico que sirve para ilustrar el diseño elegido.

## Distrito

**Organización**

Archivo de bloque con Registros de datos de longitud variable (RLV) guardados en un Hash de dispersión fija con zona de desborde. La zona de desborde será un archivo secuencial de RLV organizado por bloques para poder direccionar a otra cubeta.

**Razones**

* Ninguna funcionalidad de la aplicación requiere recorrer distritos secuencialmente una vez que encontramos a uno.
* La cantidad de distritos (al menos en un escenario real) es fija, no aumenta. Solo se harán operaciones de consulta una vez que la carga inicial fue realizada.
  + De esta forma es posible diseñar una función de dispersión así como el tamaño de los buckets para poder recuperar un distrito en la primera lectura a disco (2da en caso de que la función de hash no sea lo suficientemente efectiva y se produzca un desborde).

**Características Particulares**

* Función de Hash: A determinar.
* Tamaño Bucket/Bloque: A determinar.
* Factor empaquetamiento (registros ocupados / espacio disponible): A determinar.
* El bloque se considera desbordado cuando se algún distrito no pueda insertarse por tener tamaño mayor al espacio libre disponible. Sin embargo, si otro distrito es insertado y puede ser acomodado en el archivo de datos, el mismo puede ser dado de alta allí.

**Registros**

Se utilizara solo un registro de datos, de la forma:

* **R**(longitudRegistro,(distrito)i)

**Bloques**

Los bloques del archivo de datos y de desborde serán de la forma:

* **B**(bloqueDesborde, espacioOcupado, (Distrito)\*)

**Ilustración**

A continuación se muestra el posible contenido de un bloque:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Bloque Desborde | Espacio Ocupado | Distrito 1 | | Distrito 2 | | Distrito 3 | | Distrito 4 | |
| **4** | 42 | 5 | Chaco | 9 | Catamarca | 7 | Córdoba | 6 | Chubut |

Espacio ocupado: 4 (bloque desborde) + 4 (espacio ocupado) + 8 (longitud registro) + (5 + 9 +7 + 6) = 42

## Votante

**Organización**

Archivo de bloque con Registros de datos de longitud variable (RLV) guardados en un Hash de dispersión fija con zona de desborde. La zona de desborde será un archivo secuencial de RLV organizado por bloques para poder direccionar a otra cubeta.

**Razones**

* Ninguna funcionalidad de la aplicación requiere recorrer distritos secuencialmente una vez que encontramos a uno.
* Es necesario, para poder proveer una buena experiencia de usuario al electorado, poder acceder a la información de un votante en la menor cantidad de accesos a disco posible y de forma aleatoria.
* Una función de hashing, y tamaño de bloque (tanto de datos como de desborde) correctamente elegidos pueden hacer que en el momento más crítico (el día de las elecciones) la cantidad de lecturas necesarias para acceder a un votante sea baja. Veamos esto con un poco más de detalle:
  + Durante una elección, si un bloque se desborda debido al crecimiento de un registro, solamente se necesitaría de otro bloque para acceder a un votante. Es decir, que en la gran mayoría de los casos, como máximo se necesitarían 2 accesos a disco.
  + Una vez que las elecciones finalizan, se puede evaluar el estado del hash para determinar si una re-estructuración de mantenimiento es conveniente (en base a la cantidad de bloques con desborde).

**Características Particulares**

* Función de Hash: Considerando que el DNI de un argentino tiene 8 dígitos, tomamos los últimos 6 para determinar el bloque a donde se debe guardar el votante.
* Tamaño Bucket/Bloque: 16 KB.
  + Teniendo en cuenta que el padrón electoral argentino en la última elección fue de 28000000 votantes usaremos bloques que puedan guardar aproximadamente 28 registros.
  + A partir del tamaño del [tamaño promedio del bloque](#votanteblockav) y que debemos usar el valor más cercano hacia arriba de 512 \* 2^n.
* Factor empaquetamiento (registros ocupados / espacio disponible): A determinar (debería ser cercano a 100%).
* El bloque se considera desbordado cuando se algún votante no pueda insertarse por tener tamaño mayor al espacio libre disponible o alguna actualización haga que el bloque se desborde.

**Registros**

Se utilizara solo un registro de datos, de la forma:

* Votante (longitudRegistro, longitudDNI, (DNI)i, longitudNyA, NombreyApellido, longitudClave, clave, domicilio(longitudCalle, calle, longitudNumero, nro), longitudDistrito, (distrito)ie, ((eleccion(fecha(longitudAño, año,longitudDia, dia, longitudMes, mes), longitudCargo, cargo))ie)\*)

|  |
| --- |
| **Nota:** La longitud de algunos campos constantes tales como la clave o el DNI no es necesaria, pero sirve para usar una única implementación de RLV. De esta forma se disminuye el trabajo necesario para llevar a cabo esta organización. |

El campo de la longitud del registro es de 2 bytes. Los campos de control de cada campo son de 1 byte.

**Tamaño promedio registro**

Para calcular el tamaño promedio de un registro consideramos la siguiente longitud de sus campos.

* Control: 12 bytes.
* DNI: 4 bytes.
* Nombre y Apellido: En promedio consideramos 25 bytes.
* Clave: 4 bytes. La clave será 4 caracteres cualesquiera. Tomamos como ejemplo la clave bancaria.
* Calle: En promedio, 10 bytes.
* Numero: 2 bytes.
* Distrito: En promedio, 10 bytes.
* Fecha: 4 bytes.
* Cargo: En promedio, 10 bytes.
* Elecciones promedio: En el DNI hay 32 lugares para anotaciones. Consideramos un adicional del 25% ya que muchas veces se da la situación que estos lugares se llenan. Aproximadamente 20 elecciones como promedio, considerando que hay gente de todas las edades votando.

El tamaño promedio de un registro sería de 347 bytes.

**Bloques**

Los bloques del archivo de datos y de desborde serán de la forma:

* **B**(bloqueDesborde, espacioOcupado, (Votante)\*)

De esta forma el tamaño promedio de bloque será

28 votantes \* 347 bytes + 4 bytes + 2 bytes = 9722 bytes.