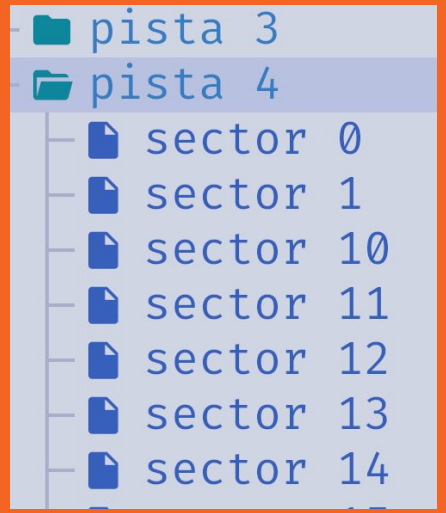
SGBD

Jose Javier Alva Cornejo

Estructura de disco

```
═══ Crear Disco ═══
Nombre de disco a crear: disco
Ingrese número de superficies: 6
Ingrese número de pistas por superficie: 20
Ingrese número de sectores por pista: 20
Ingrese número de bytes por sector: 512
Ingrese número de sectores por bloque: 4

Disco creado exitosamente

ENTER para regresar ...
```



```
// Se crea toda la estructura                                    static library(disk_manager
for (int s = 0; s < SURFACES; ++s) {
    std::string carpetaSuperficie = disk_name_used + "/superficie " + std::to_string(s);
    fs::create_directory(carpetaSuperficie);

    for (int p = 0; p < TRACKS_PER_SURFACE; ++p) {
        std::string carpetaPista = carpetaSuperficie + "/pista " + std::to_string(p);
        fs::create_directory(carpetaPista);

        for (int sec = 0; sec < SECTORS_PER_TRACK; ++sec) {
            std::string archivoSector = carpetaPista + "/sector " + std::to_string(sec);

            if (bin) {
                std::ofstream out(archivoSector, std::ios::binary | std::ios::trunc);
                std::vector<char> zeros(SECTOR_SIZE, 0);
                out.write(zeros.data(), SECTOR_SIZE);
                out.close();
            } else {
                std::ofstream(archivoSector + ".txt").close();
            }
        }
    }
}
```

Sector 0


```
Superficies: 6, tracks: 20, sectores: 20,
tamanio de sector: 512,sectores por
bloque: 4--
```

```
/*
 * Representa metadata esencial de un disco
 */
#pragma pack(push, 1)
struct Sector_0 {
  uint16_t surfaces{};
  uint16_t tracks_per_surf{};
  uint16_t sectors_per_track{};
  uint16_t sector_size{};
  uint16_t sectors_per_block{};
};
#pragma pack(pop)
```

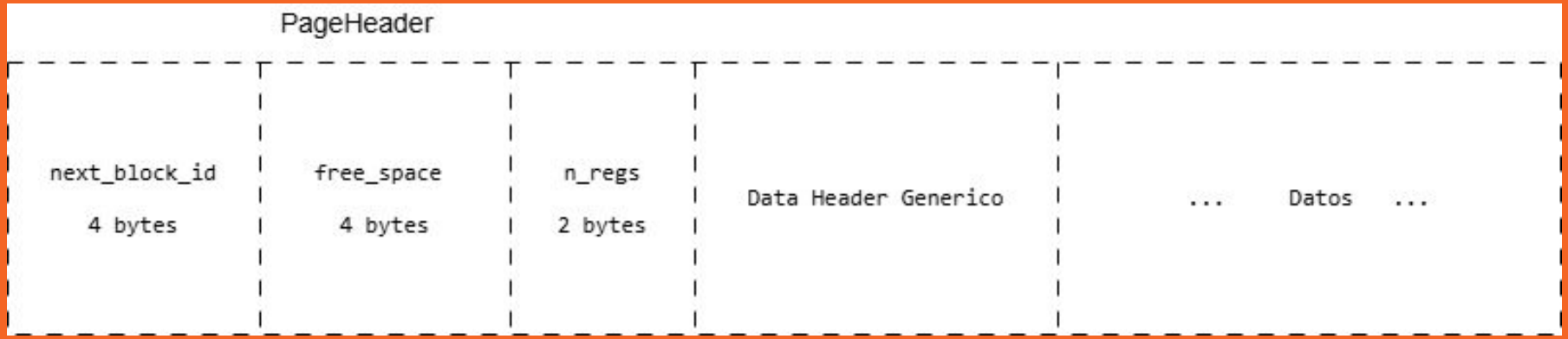
Sector 1


```
0titanic
Max_reg_size: 216
Fixed_regs: 1
First page id: 1
Last page id: 99
n_columns 12
PassengerId 2 4-
Survived 2 4-
Pclass 2 4-
Name 6 100-
Sex 6 20-
Age 5 8-
SibSp 2 4-
Parch 2 4-
Ticket 6 20-
Fare 5 8-
Cabin 6 20-
Embarked 6 20-
```

```
/*
 * Representa las ubicaciones de todas las relaciones
 * en disco(sector reservado 1)
 * A cada tabla le corresponde un bloque para
 * guardado de metadata
 */
struct Sector_1 {
  uint8_t n_tables{};
  std::vector<uint32_t> table_block_ids{};
};
```

Estructura de paginas

PageHeader

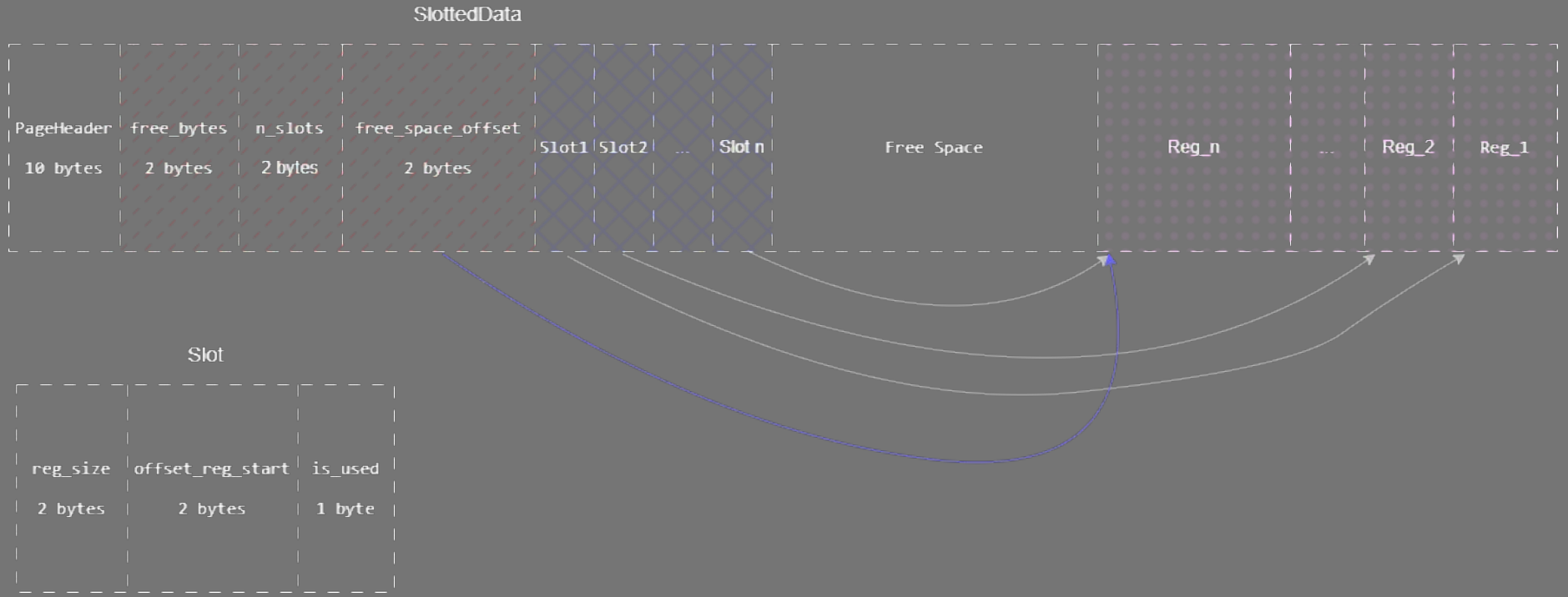

PageHeader
next_block_id
4 bytes
free_space
4 bytes
n_regs
2 bytes
Data Header Generico
... Datos ...

```
#pragma pack(push, 1)
struct PageHeader {
    uint32_t next_block_id{};
    uint32_t free_space{};
    uint16_t n_regs{};
};
#pragma pack(pop)
```

FixedDataHeader

```cpp
/*
 * @struct Estructura para metadata de bloques interno de file donde
 * registros son de size fijos
 * @note El bitset tiene tamanio de la max cantidad de registros guardables
 */
struct FixedDataHeader {
  // Respecto a bloque completo, se resta tamanio mismo de header
  uint32_t free_bytes{};
  uint32_t reg_size{};
  uint16_t max_n_regs{};
  boost::dynamic_bitset<unsigned char>
      free_register_bitmap;
};
```

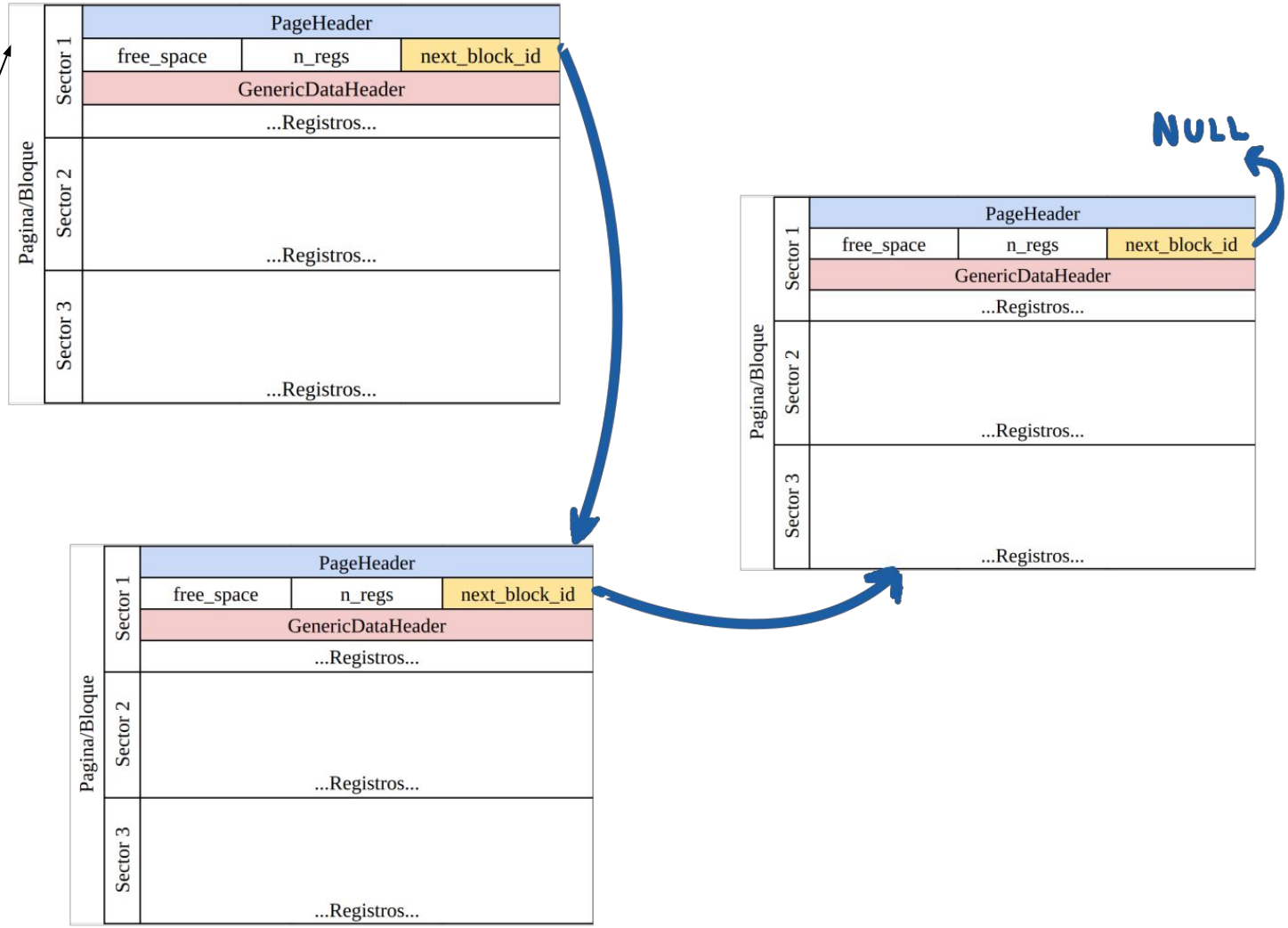
SlottedDataHeader



Ejemplo de serialización

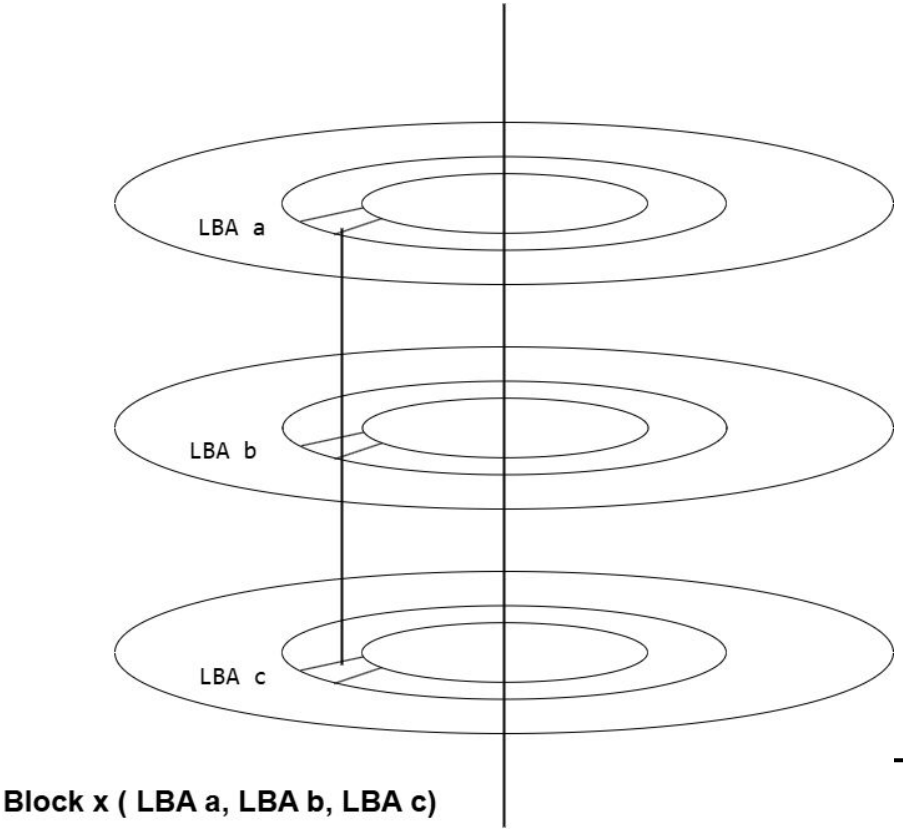
```cpp
template <typename Iter>
inline void serialize_fixed_block_header(const FixedDataHeader &header, Iter &out_it) {
  write_v(out_it, header.free_bytes);
  write_v(out_it, header.reg_size);
  write_v(out_it, header.max_n_regs);

  size_t bitset_byte_size = (header.max_n_regs + CHAR_BIT - 1) / CHAR_BIT;

  boost::to_block_range(b: header.free_register_bitmap, result: out_it);
  std::advance(out_it, bitset_byte_size);
}
```

Tabla First_page_id



Bloques

Diccionario de datos/FreeBlockMap



Asignacion vertical

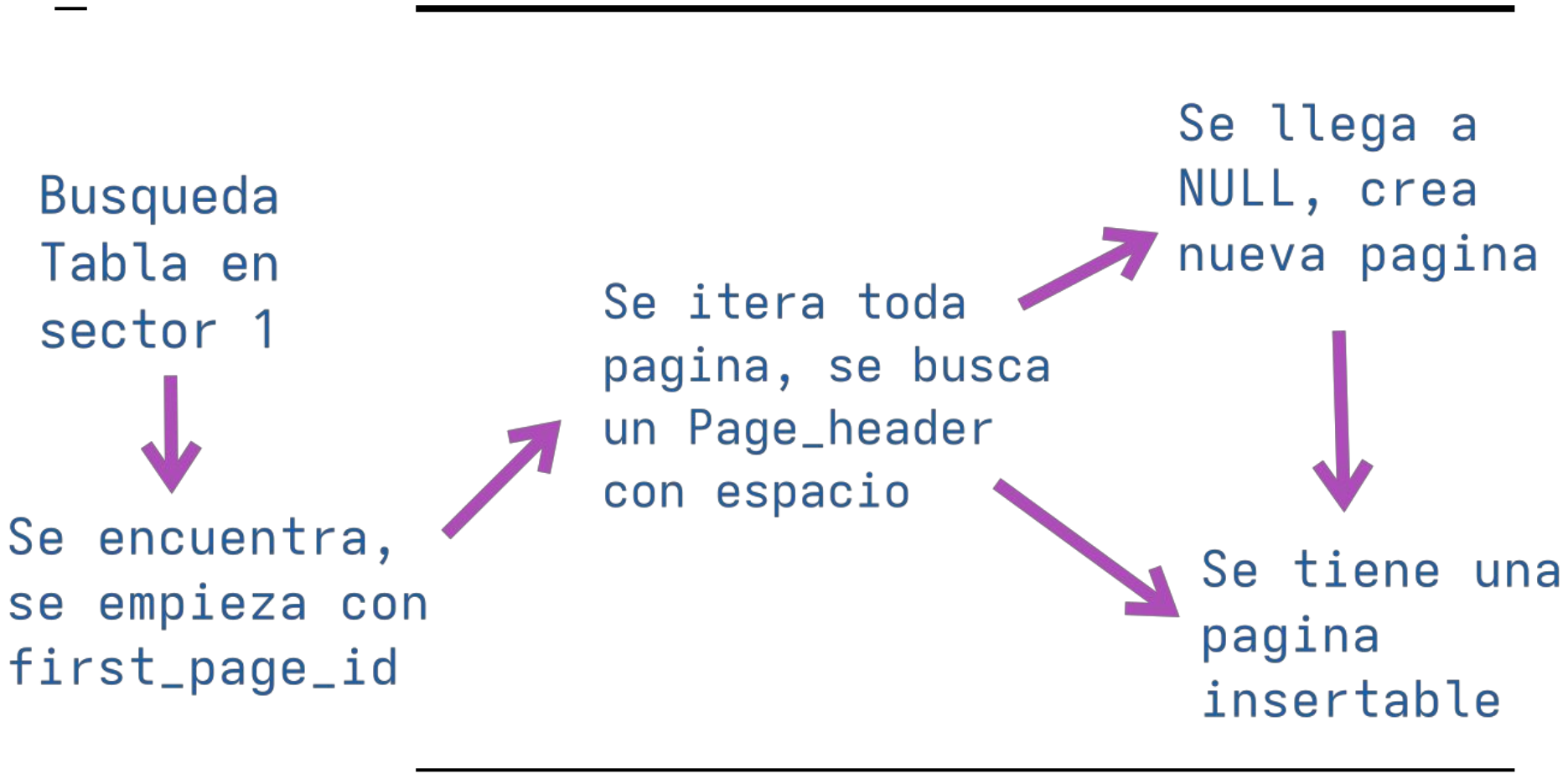
Bloque agrupa ciertos sectores

```cpp
struct FreeBlockMap {
    std::vector<std::pair<bool, std::vector<uint32_t>>> blocks{};

    bool is_block_free(uint32_t block_id) {
        if (block_id >= blocks.size())
            throw std::runtime_error("block_id fuera de rango");

        return !blocks[block_id].first;
    }

    uint32_t get_ith_lba(uint32_t block_id, size_t ith_lba) {
        if (block_id >= blocks.size())
            throw std::runtime_error("block_id fuera de rango");

        return blocks[block_id].second[ith_lba];
    }

    void set_block_used(uint32_t block_id) {
        if (block_id >= blocks.size())
            throw std::runtime_error("block_id fuera de rango");

        blocks[block_id].first = true;
    }
};
```

```
Next_page_id: 2 N_registers: 18 Free_space/capacity: 4/404
free_space_offset: 110 # Slots: 18
 |1|99|1901|
|1|128|1773|
|1|107|1666|
|1|119|1547|
|1|97|1450|
|1|89|1361|
```

```
Superficie: 3 Track: 0 Sector: 1
103"Braund; Mr. Owen Harris"male22.00000010A/5 211717.250000                    S
211"Cumings; Mrs. John Bradley (Florence Briggs Thayer)"female38.00000010PC 1759971.
283300C85                C
313"Heikkinen; Miss. Laina"female26.00000000STON/O2. 31012827.925000                  S
411"Futrelle; Mrs. Jacques Heath (Lily May Peel)"female35.0000001011380353.
100000C123               S

Superficie: 2 Track: 0 Sector: 1
503"Allen; Mr. William Henry"male35.000000003734508.050000                    S
603"Moran; Mr. James"male0.000000003308778.458300                  Q
701"McCarthy; Mr. Timothy J"male54.000000001746351.862500E46                 S
803"Palsson; Master. Gosta Leonard"male2.0000003134990921.075000                   S
913"Johnson; Mrs. Oscar W (Elisabeth Vilhelmina Berg)"female27.0000000234774211.
133300                   S

Superficie: 1 Track: 0 Sector: 1
1012"Nasser; Mrs. Nicholas (Adele Achem)"female14.0000001023773630.070800
```

Manejo de Registros


Busqueda Tabla en sector 1
Se encuentra, se empieza con first_page_id
Se itera toda pagina, se busca un Page_header con espacio
Se llega a NULL, crea nueva pagina
Se tiene una pagina insertable

Inserción de un registro

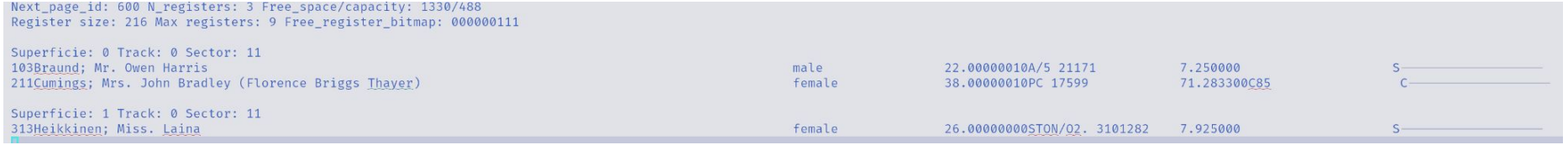
```
Next_page_id: 600 N_registers: 1 Free_space/capacity: 1762/488
Register size: 216 Max registers: 9 Free_register_bitmap: 000000001

Superficie: 0 Track: 0 Sector: 11
103Braund; Mr. Owen
Harris
male                22.00000010A/5 21171          7.250000
S


Superficie: 1 Track: 0 Sector: 11
```

Linea “wrappeada” por motivos de visualizar, en realidad es una linea continua

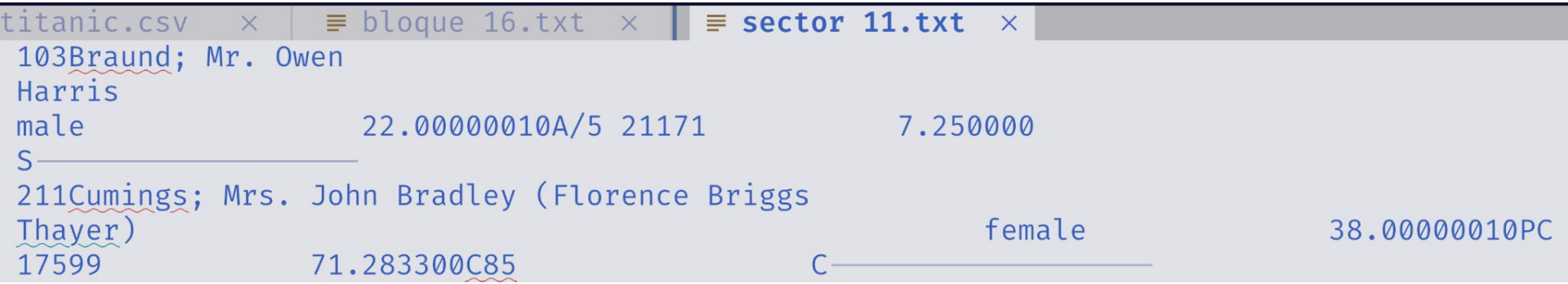
titanic.csv × bloque 16.txt × **sector 11.txt** ×

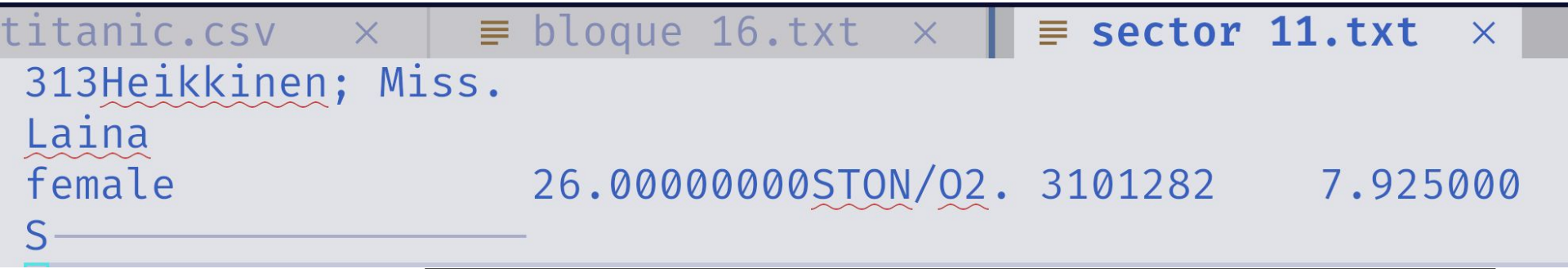
```
103Braund; Mr. Owen
Harris
male                22.00000010A/5 21171          7.250000
S
```

Inserción hasta exceder un sector




```
Next_page_id: 600 N_registers: 3 Free_space/capacity: 1330/488
Register size: 216 Max registers: 9 Free_register_bitmap: 000000111

Superficie: 0 Track: 0 Sector: 11
103Braund; Mr. Owen
Harris
        male                22.00000010A/5 21171           7.
250000                    S
211Cumings; Mr. John Bradley (Florence Briggs
Thayer)
female              38.00000010PC 17599           71.
283300C85                 C

Superficie: 1 Track: 0 Sector: 11
313Heikkinen; Miss.
Laina
        female              26.00000000STON/O2. 3101282    7.
925000                    S
```


titanic.csv
bloque 16.txt
sector 11.txt
103Braund; Mr. Owen
Harris
male 22.00000010A/5 21171 7.250000
S
211Cumings; Mrs. John Bradley (Florence Briggs
Thayer)
female 38.00000010PC
17599 71.283300C85 C


titanic.csv
bloque 16.txt
sector 11.txt
313Heikkinen; Miss.
Laina
female 26.00000000STON/O2. 3101282 7.925000
S

Inserción de un registro variable

```
Next_page_id: 600 N_registers: 1 Free_space/capacity: 1851/489
free_space_offset: 1872 # Slots: 1
 |1|128|1872|
|
Superficie: 3 Track: 0 Sector: 1
1903Vander Planke; Mrs. Julius (Emelia Maria Vandemoortele)female31.0000001034576318.
000000                    S
```

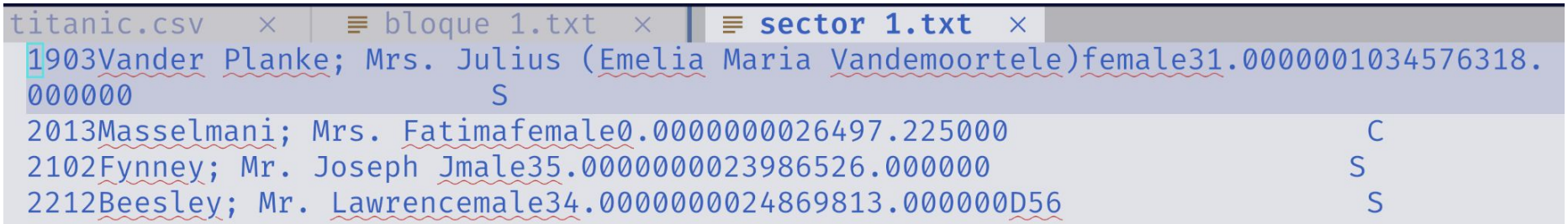
titanic.csv × | bloque 1.txt × | **sector 1.txt** ×

```
1903Vander Planke; Mrs. Julius (Emelia Maria Vandemoortele)female31.0000001034576318.
000000                    S
```

Inserción variable hasta exceder sector

titanic.csv × | bloque 1.txt ×

```
Next_page_id: 600 N_registers: 5 Free_space/capacity: 1456/469
free_space_offset: 1497 # Slots: 5
 |1|128|1872|
|1|94|1778|
|1|91|1687|
|1|92|1595|
|1|98|1497|
|
Superficie: 3 Track: 0 Sector: 1
1903Vander Planke; Mrs. Julius (Emelia Maria Vandemoortele)female31.0000001034576318.
000000                      S
2013Masselmani; Mrs. Fatimafemale0.0000000026497.225000                        C
2102Fynney; Mr. Joseph Jmale35.0000000023986526.000000                        S
2212Beesley; Mr. Lawrencemale34.0000000024869813.000000D56                     S

Superficie: 2 Track: 0 Sector: 1
2313McGowan; Miss. Anna Anniefemale15.000000003309238.029200                      Q
```


titanic.csv
bloque 1.txt
sector 1.txt
1903Vander Planke; Mrs. Julius (Emelia Maria Vandemoortele)female31.0000001034576318.
000000 S
2013Masselmani; Mrs. Fatimafemale0.0000000026497.225000 C
2102Fynney; Mr. Joseph Jmale35.0000000023986526.000000 S
2212Beesley; Mr. Lawrencemale34.0000000024869813.000000D56 S


titanic.csv
bloque 1.txt
sector 1.txt
2313McGowan; Miss. Anna Anniefemale15.000000003309238.029200 Q

Insert Fixed implementación

```
// Serializamos todo el registro                                    static library(megatron)
auto register_bytes: std::vector<unsigned char> = serialize_register(table_metadata,
                                                  &: values);
```

```
// Se busca pagina a insertar
std::vector<unsigned char> page;
uint32_t insert_page_id;

insert_page_id = get_insertable_page(&: page,
                                     block_id: table_metadata.first_page_id,
                                     reg_size: table_metadata.max_reg_size);

// Paginas sin espacio suficiente
if (insert_page_id == disk.NULL_BLOCK)
  insert_page_id = add_new_page_to_table(&: table_metadata);
```

```
// Calculamos posicion donde insertar
size_t free_reg_pos = serial::find_free_reg_pos(header: fixed_data_header);
byte_offset_free_reg = serial::calculate_reg_offset(header: fixed_data_header,
                                                    nth_reg: free_reg_pos);
```

```
disk.write_block(&block_bytes: insert_page_bytes, block_id: insert_page_id);
```

Insert Variable Implementación

```
size_t free_slot = serial::get_free_slot(&: slotted_data_header);
if (free_slot == slotted_data_header.n_slots)
  free_slot = serial::add_free_slot(&: page_header, &: slotted_data_header);

byte_offset_free_reg = serial::insert_into_slotted(&header: slotted_data_header,
                                                   n_slot: free_slot,
                                                   reg_size: register_bytes.size());
```

Select Fixed

```
// Se saca metadata relevante                                    static library(megatron)
serial::PageHeader page_header;
serial::FixedDataHeader fixed_data_header;
serial::SlottedDataHeader slotted_data_header;

page_header = serial::deserialize_page_header(page_bytes_it);

fixed_data_header = serial::deserialize_fixed_data_header(page_bytes_it);
for (size_t i{}; i < fixed_data_header.max_n_regs; ++i) {
  if (fixed_data_header.free_register_bitmap.at(i)) { // Registro existe
    auto register_bytes = get_ith_register_bytes(table_metadata, page_header, fixed_data
    auto register_values = deserialize_register(table_metadata, register_bytes);

    // Si hay condicion
    if (col_index < table_metadata.n_cols && register_values[col_index] ≠ cond_val)
      continue;

    for (auto &v : register_values)
      std::cout << SQL_type_to_string(v) << " | ";

    std::cout << std::endl;
  }
}
```

Select Slotted

```
// Se saca metadata relevante                                    static library(megatron)
serial::PageHeader page_header;
serial::FixedDataHeader fixed_data_header;
serial::SlottedDataHeader slotted_data_header;

page_header = serial::deserialize_page_header(page_bytes_it);

slotted_data_header = serial::deserialize_slotted_data_header(page_bytes_it);
for (size_t i{}; i < slotted_data_header.n_slots; ++i) {
  if (slotted_data_header.slots[i].is_used) { // Registro existe
    auto register_bytes = get_ith_register_bytes(table_metadata, page_header,
    |                                            slotted_data_header, page_bytes, i);
    auto register_values = deserialize_register(table_metadata, register_bytes);

    // Si hay condicion
    if (col_index < table_metadata.n_cols && register_values[col_index] != cond_val)
      continue;

    for (auto &v : register_values)
      std::cout << SQL_type_to_string(v) << " | ";

    std::cout << std::endl;
  }
}
```

Eliminar fijo -> Similar a select

```
if (register_values[col_index] ≠ cond_val)
    continue;

// Si cumple condicion, delete
// Solo marcamos como libre, ya que todo es fijo se reescribira luego
fixed_data_header.free_bytes += fixed_data_header.reg_size;
fixed_data_header.free_register_bitmap[i] = false;
page_header.free_space += fixed_data_header.reg_size;
page_header.n_regs--;
```

Eliminar variable

```
if (register_values[col_index] ≠ cond_val)
    continue;

// Cumple condicion, solo marcamos slot como disponible
// No se actualiza free_space, este depende de un compactar
slotted_data_header.slots[i].is_used = false;
page_header.n_regs--;
```

Luego de delete PClass 3

```
Next_page_id: 2 N_registers: 6 Free_space/capacity: 4/404
free_space_offset: 110 # Slots: 18
 |0|99|1901|
|1|128|1773|
|0|107|1666|
|1|119|1547|
|0|97|1450|
|0|89|1361|
|1|95|1266|
|0|103|1163|
|0|124|1039|
|1|110|929|
|0|107|822|
|1|99|723|
|0|106|617|
|0|100|517|
|0|111|406|
|1|107|299|
|0|93|206|
|0|96|110|
```

```
Superficie: 3 Track: 0 Sector: 1


211"Cumings; Mrs. John Bradley (Florence Briggs Thayer)"female38.00000010PC 1759971.
283300C85                  C


411"Futrelle; Mrs. Jacques Heath (Lily May Peel)"female35.0000001011380353.
100000C123                 S

Superficie: 2 Track: 0 Sector: 1
```

Referencias

1. Silberschatz, A., Korth, H. F., & Sudarshan, S. (2011). Database system concepts (6th ed.). McGraw-Hill.
2. Garcia-Molina, H., Ullman, J. D., & Widom, J. (2008). Database systems: The complete book (2nd ed.). Pearson Education.
3.