**File hierarchy**

store-data

living-store-serialisation.bin // growing list of actions to reconstitute in-memory living store

front-store-location.state // contains store1 or store2

store1 // front-store or back-store

part000.bin

part001.bin

part002.bin

store2 // back-store or front-store

part000.bin // fusion des 3 parties en cours de construction

**Format de partxxx.bin**

Il contient une image figée d'une hashtable contenant les couples (occurrenceCount, fileID) pour chaque mot. Pour accéder au bon bucket à partir d'un certain mot, il suffit de prendre le hash de ce mot modulo le nombre de bucket.

Dans la description du format, les éléments d'une table font tous la même taille, contrairement à ceux d'une liste qui peuvent être de taille variable.

Format :

* Nombre de buckets **b** (long)
* Table de **b** addresses de buckets
  + Adresse d'un bucket dans le fichier (long)
* Liste de **b** buckets triès par
  + *Chaque Bucket commence à l'adresse associée indiquée dans la table précédante*
  + Nombre de clefs **e** dans ce bucket (int)
  + Liste de **e** clefs
    - Mot (StringUTF max 100 carractères)
    - Adresse d'un WordBlock dans le fichier (long)
* Liste de tous lesWordBlock, triée par mot
  + *Chaque WordBlock commence à l'adresse indiquée par la clef associée*
  + Mot (StringUTF max 100 carractères)
  + Nombre total d'occurrence du mot pour cette Part (long, peut être négatif)
  + Nombre de couples **c** (long)
  + Table de **c** couples pour ce mot, triés avec l'ordre décrit plus bas
    - Nombre partiel d'occurrences (long, peut être négatif)
    - ID du fichier du drive, le même que celui dans la base de donnée (long)

L'ordre des éléments dans un WordBlock est le suivant :

public int compareTo(Element other) {

long absPartialCount1 = Math.abs(this.partialCount);

boolean deletionFlag1 = this.partialCount < 0;

long fileID1 = this.fileID;

long absPartialCount2 = Math.abs(other.partialCount);

boolean deletionFlag2 = other.partialCount < 0;

long fileID2 = other.fileID;

if(absPartialCount1 != absPartialCount2) {

return -Long.compare(absPartialCount1, absPartialCount2);

}

if(fileID1 != fileID2) {

return -Long.compare(fileID1, fileID2);

}

return -Boolean.compare(deletionFlag1, deletionFlag2);

}

Cet ordre est défini de sorte à ce qu'un élément de deletion (partialCount négatif) suive imédiatement l'élément d'insertion correspondant (possiblement dans une autre Part)