# Sinphinity Documentation

## Representation of Music Information

We start with midi files but we convert it to a different representation. Midi format is OK if all you want to do is to play the music, but it is bad for analysis of the music structure.

Our representation consists of the following elements:

1. An array of notes. For each Note ON-Note OFF element that we find in the Midi file, we create a note object. The note object has the time information in ticks from the beginning of the song, not ticks from the previous event. The note object has also the pitch, the volume, the instrument, the voice (that is a concept similar to the midi track)
2. An array of bars. Each bar object has a time signature, a bar number (that starts with 1) and the ticks from the beginning of the song
3. An array of tempo changes. We don’t pay much attention to the tempo when analysing a song. The tempo doesn’t change the structure of the song.

We can convert back and forth between this format and the Midi format. When we want to play music, we use the Midi format, when we want to analyse it, we use our format.

### Cleaning

When we create the objects for our representation of a song, we make some changes to the original version, that makes the analysis simpler. This is mainly because what the musician plays is not exactly what the score says. For example a note may be played 3 ticks after the beat, but really it was supposed to be played exactly in the beat. These “small errors” are not perceived by the ear most of the time, but if we leave them, the analysis becomes more complex.

### Simplifications

So we keep a version of the song that is exactly what the Midi file has, and we called “simplification version 0). We then do this “cleaning” (where for ex. we also remove small silences between notes, or shorten notes that extend a little bit after the beginning of the next note). This “cleaned” version is called “simplification version 1”. We may in the future do more “cleaning” like removing ornaments, and call it “simplification version 2”, etc.

### Voices

Another thing we do when converting from midi to our representation is to separate different voices that may be present in the same track. For ex. a piano classical peace, that has a right hand playing one voice and a left hand playing a different voice, usually has all the notes in the same track in the midi representation. We need to separate them in 2 different groups of notes to do the analysis. So we use the concept of voice, instead of tracks. A track may contain independent melodies, a voice has 1 single melody.

When we have a track that is playing mostly chords (that is, it has polyphony, but the notes that sound together start and end together), we don’t separate the notes in different voices.

### Phrases

As part of the analysis of the songs data, we extract the phrases in each song. A phrase is part of a melody. We ignore the duration of the notes when extracting phrases, we care only about the time of the start of the note and the pitch. So if a note is played until the start of the next note, or if it is played staccato way (ending before the next note starts) the phrase is the same. We also don’t care about the “absolute pitch” but the “relative pitch”, where we call relative pitch to the difference between 2 consecutive pitches. So for ex a phrase with the 3 notes C,D,E, is the same as F,G,A, since the distance in pitches between the notes is the same.

Making these assumptions, we can put in the same bag 2 phrases that are essentially the same, even if they are played with different style or transposed.

There is an important decision to make when extracting phrases, that is where to cut the song and say here is the end of a phrase and starts the next. We consider these “break points” to be:

- when there is a long silence between 2 consecutive notes in a melody or there is a long note (long compared with the other notes in the vicinity)

- when there are many consecutive notes that are all of the same duration (lets’ say 32 consecutive sixteens), we break at the beginning of the bars in such a way that a phrase is never longer than 2 bars and not shorter than half a bar.

It is useful to analyse the metric structure of the phrases and the pitches structure separately, so we can for example find 2 phrases in a song that have the same metric but with slightly different pitches, where the variation of the pitches is part of the reason they work well in the song.

When analysing the metrics of a phrase, a succession of 8 consecutive quarters is not much different of a succession of 8 consecutive eights: they are 8 equally spaced notes. So we consider also the “basic metrics” of phrases, where we don’t care about the absolute duration of the notes, but only their relative durations. So in the case of the 8 consecutive quarters, we have 8 ones, and if we have for ex. 4 quarters followed by 4 eights, we would have 4 twos followed by 4 ones.

It is also useful to remove ornaments when looking for phrases, because the ornaments may give it an interesting flavour to a phrase, but it is still the same phrase.

In order to see how phrases are used by different composers in different songs, it is useful to save a “bookmark” wherever the phrase was used.

So, when analysing the phrases of a song we extract the following:

* An array of basic metrics
* An array of phrases metrics
* An array of phrases pitches
* An array of phrases (that have metrics and pitches)
* An array of embellished metrics (where we didn’t remove the ornaments)
* An array of embellished pitches (where we didn’t remove the ornaments)
* An array of embellished phrases (where we didn’t remove the ornaments)
* An array of Phrases Occurrences, with the information of where a phrase was used in a song

### Database storing

When we have the music information in RAM we use C# classes like arrays of notes and arrays of bars. To persist the data we opted for SQL (SQL Server right now), that is well known and powerful. To do the interesting stuff we use Graph Databases, but we store everything initially to SQL and from there to the Graph Db.

In SQL we use a table for songs, that has the name and a foreign key to the band. It has also information about the processing status of the song. There is a table MidiStats, linked to this, that has information like the duration of the song, the number of tracks, etc. Another table, called SongsData, also linked to Songs, has the original midi file content (encoded in Base64), the array of bars and the array of tempo changes. These arrays are saved as json. And finally there is a SongSimplifications table, linked to SongsData, that has the array of notes of a simplification of the song. There is 1 record for each simplification of each song.