== 1. Overview

¶ At dharmalekha.info
Let me first give you a brief overview of what we are doing in the DHARMA project. We have a website at dharmalekha.info.
~50 contributors. I am counting all people who are credited in editions. In practice, less people than that are actually encoding texts.
~3,300 texts in 13+ languages. The immense majority of these texts are inscriptions. But we also have editions of manuscripts (about 50 so far).

Texts are hosted on github, in various repositories. We also have a project-wide bibliography on zotero.org.

In terms of space, the project's database is tiny: it weighs ~150 Mebibytes. This includes lookup tables, of course. But also a full copy of all our inscriptions, and a full copy of our bibliography.

¶ Data sources
Our editions come from various sources.

We have original editions (I mean, editions that are not available elsewhere). But we are also incorporating editions from elsewhere. The number of texts is thus rapidly growing.

We are incorporating digital editions from other projects. Thus, the Early Inscriptions of Āndhradeśa (EIAD) and SIDDHAM projects.

We are also digitizing printed editions. For instance, we will progressively add inscriptions from the South Indian Inscriptions series. About ~12,000 inscriptions have been digitized by Thomas Malten. I will convert them to the proper format in due time.

I have not done much digitization work so far. It is my predecessor, Axelle Janiak, who did almost everything.

¶ Languages distribution
It might be interesting to take a look at languages distribution. The percentages you are seeing on the slide take into account the length of each text. Conceptually, I concatenated all editions into a single huge text and counted how much of it is Sanskrit, how much of it is Tamil, and so on.

For now, Sanskrit predominates, but this will probably change as time goes on. The 12,000 South Indian Inscriptions I alluded to earlier will be in Tamil.

It is important to note that we have many multilingual texts; for instance, Tamil texts with passages in Sanskrit.

¶ Where are we at?
I have been working for DHARMA for 1.5 year. So far, I worked on the infrastructure and the display system.

* Infrastructure. I mean the underlying logic. Essentially, the server setup and the database update system.
  Mostly OK, but I made mistakes that need to be addressed. That's why you have a bug on the slide.
* Display. I mean generating web pages for the editions and the bibliography, among other things.
  Mostly done. There are bugs in there. That's why you have a ladybug on the slide.
* Search. For now, we have nothing. I plan to start working on it as soon as possible (2 or 3 months). Indeed, it is by far the most complicated task. We have 2 years to do that, which is not much, considering the project started in 2019. I believe my predecessor and I have spent way too much time on display-related issues already.

Let us review these three points in order. I will say a few words about each them.

== 2. Infrastructure

Some words on the infrastructure.

¶ Which Database?
For projects like DHARMA, people often use eXistdb. Despite the name, it's not really a database, but a complete framework for digital editions. It has a web server, an online XML editor, and fancy stuff like that.

I advocated against it, for many reasons. Briefly put, existDB is one of those tools that make the easy stuff easier but also makes the complicated stuff more complicated. It wants you to do everything in XQuery, which is an abomination of a language. You can't write real programs with it, unless you like torture.

I advocated for SQLite instead. It is the most reliable database system I have found so far. It is reliable in the sense that it is extremely well-tested. But also because the author has a strict policy for backwards-compatibility. In other words, it won't break your code. This is important for long-term projects.

SQLite is also much smaller (1 MiB of compiled code on a 64-bits architecture).

That said, SQLite is not a panacea. It does less, so you have to write more code.

¶ Concurrency
SQLite also has an important limitation: you can't have concurrent write transactions. It is your job, as a developer, to make sure this doesn't happen. People generally use concurrency primitives (threads and mutexes) to address this. I am lazy, so I opted for the simplest method, which is to have a single writer process do all the work.

Furthermore, I have taken some dispositions to avoid race conditions. Firstly, the writer process is completely independent from other database clients. They don't communicate with each other and they don't share mutable data. Secondly, the writer process is the only process that is allowed to access git repositories on the filesystem. Reader processes talk to the database instead. For this reason, I store there a copy of all the files reader processes might need to access.

Briefly put, I am delegating all the concurrency handling to SQLite instead of doing it myself.

¶ Update system
Now let us look at the update system. It is very simple, as you can see. Arrows represent messages. When someone pushes to a github repository, our HTTP server is notified, and it notifies in turn the writer process, which will do all the work necessary to update the database: it pulls the repository, figures out which files have been modified/added/deleted, validates the files against an XML schema, updates the database accordingly, and so on.

Idem for the bibliography.

In addition, the writer process schedules a full update (all repositories + bibliography) every few hours. Thus, if the message queue overflows and loses messages, the database will still be up-to-date in the near future. In the worst case, if the writer process is so slow that it can't process messages, it will just end up running a series of full updates.

== 3. Display

¶ Display: demo
Now some words on the display. Let us look at the Website.

There are two points of interest for people outside DHARMA: "texts" and "parallels".

Let us look at the "texts" interface. https://dharmalekha.info/texts
We have a basic search interface. It's not meant to provide "real" search capabilities. The goal is just to allow you to filter texts by editor, language, repository, etc. I will replace it with something better later on.

Display example: https://dharmalekha.info/texts/INSPallava00256. There are three displays: logical, physical, full. Arguably, "full" should now be the default. It shows all the information encoded in editions, hence its name. "logical" only shows the corrected text (with emendations, spelling normalization, etc.). And "physical" shows the original text (without emendations, etc.).

Next we have a tool for searching approximate parallels of metrical passages (quarter verses, half-verses and verses). I initially wrote that to help Amandine Wattelier-Bricout, one of our Post-Docs. Currently, there are two lookup interfaces. You can either input something and look at the results. Or you can browse the corpus by text. The data for the latter interface is pregenerated, so lookup is fast enough. But it is generated in quadratic time, so this does not scale well. I need to improve that.

== 4. Search

¶ Search: TODO
Now a few words about search functionalities. I haven't written anything so far, so this is just a TODO list.

SQLite already has a good full-text search system. This will save me some work. However, it doesn't support faceted search, so I will have to write this functionality. EXPLAIN faceted search. SQLite also doesn't include anything related to language processing, so I will have to hook it up with external libraries.

We must distinguish searching modern Latin languages (the ones used in translation and notes, generally English) and searching source texts (in Sanskrit, Tamil and so on). For Latin languages, people have already written all needed components: lexical analyzers, in particular. The delicate part is to stick the pieces together in an effective way.

For searching source languages, things will be more complicated. Firstly, because we have many languages, with various transliteration conventions. I can do the necessary work for Sanskrit, but I don't know the other languages used in the project. In any case, we need sopmething that will work for all languages. Another difficulty is that people want to be able to search both the logical representation of inscriptions and the physical one. We can expect the "logical" representation to be "correct" (thus we could parse it, in theory), but not the "physical" one.

I thus think that, for now, we should go for the lowest common denominator. We have 5 MiB of plain text for editions. This is nothing, so let's use brute force.

¶ TRE:
I plan to use the TRE library. It is rather old (20+ years old). Suprisingly, it hasn't gotten much traction. But it is in fact really nice. It is able to do approximate search (Levenshtein distance).
* It does so in time linear to the input size

* It does so with constant space requirements (it preallocates a small stack and works with that)

I am using it on my website.