



Emulation options for legacy databases

Klaus Rechert (University Freiburg i.Br., Germany)

Klaus Rechert recently became professor at the University of Applied Sciences Kehl. In October 2021, he was researcher and professor at the University of Freiburg. He was the principle investigator of bwFLA (Baden-Württemberg Functional Long-Term Archiving and Access) and has been the architect behind Emulation as a Service. He is involved in multiple national and international projects related to digital preservation, reproducible science and research data management. He was awarded a Diploma in Computer Science in 2005, and a doctoral degree in 2013 from the University of Freiburg.

Rechert started his presentation by citing the challenge of a safe and economic preservation over many decades. He reminded the audience of a „Database“ being a weakly defined term, from a Computer Science perspective (hence a definition given by Naumann, [p. 6](#)). Is it possible to plan this far ahead and what are the risks? There are two basic strategies: either data-driven – focused on extracting the data, or software-driven – focused on the technical stack. There is also a middle ground between the two.

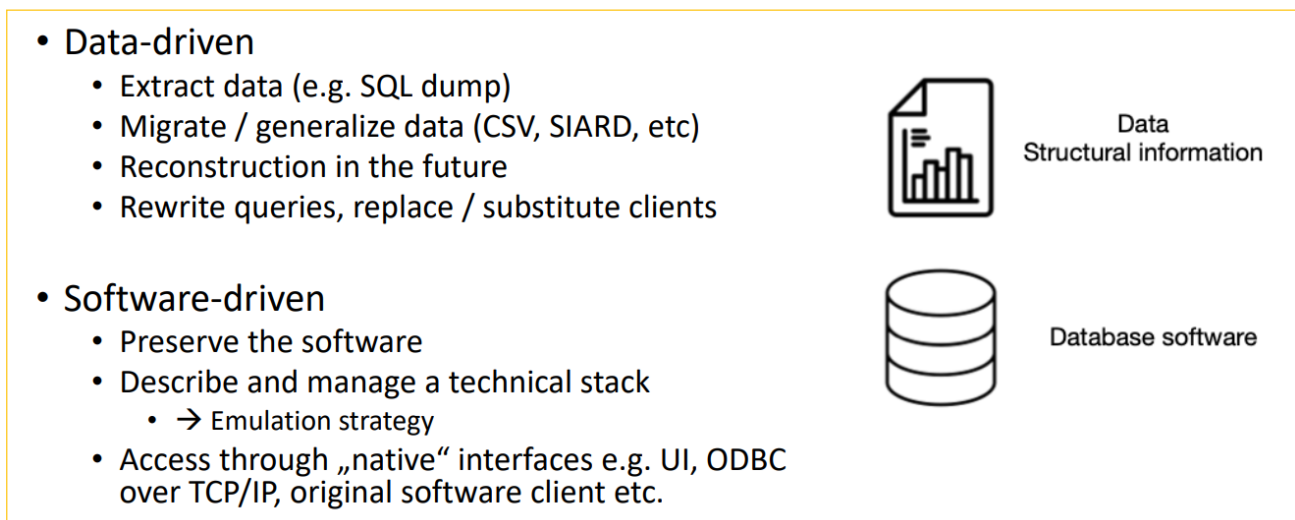


Figure 9: The data-driven and the software-driven strategy

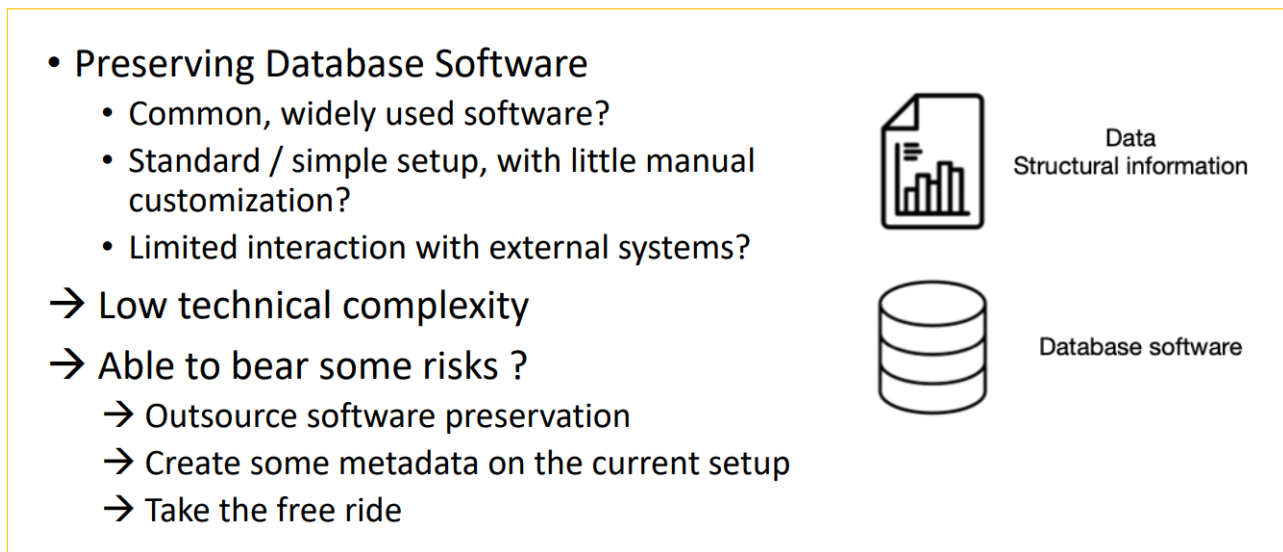


Figure 10: The (lazy) strategy

Reichert referred to a software-driven strategy that mostly relies on emulation. There are several subtypes:

- A lazy strategy – if you have a widely used database software already set up with little manual customization, you can create metadata around the dump and take a free ride, hoping for an emulation solution to be around on the day the data are revived. However, this is risky.
- A less lazy strategy is to preserve the database software and use a contemporary emulation framework. This is a better way of ensuring that it will work in the future.

The technical stack for the less lazy strategy (figure 11) comprises the:

- data,
- database software,
- configured environment,
- virtual hardware, and
- contemporary CPUs and connectors to machines and users.



- Preserving Database Software
 - Common, widely used software?
 - Standard / simple setup, with little manual customization?
 - Limited interaction with external systems?

→ Use a contemporary emulation framework

→ Create an installation of the database and deploy the data

→ Verify your result

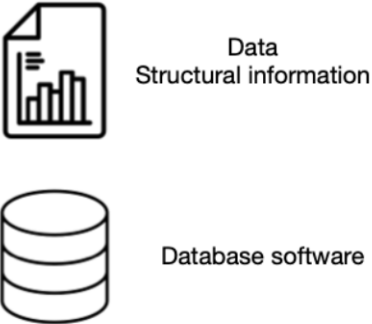


Figure 11: The (less lazy) strategy

- Emulation as Conceptual Framework
 - Different options to preserve a DB instance
 - Complexity matters
 - Long-term risk profile
 - Emulation is software-based
 - Preservation planning: prepare for obsolescence
 - Scaling emulation
 - Emulation scales well with users! Cooperate!
 - Plenty of automation options

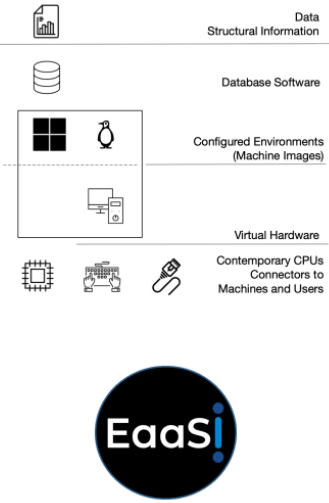


Figure 12: Conclusions on emulation

Complex setups may require manual or installation steps. If you have Microsoft Office Sharepoint and you know the version of Sharepoint, you can create a simple solution using a template, as this is a common scenario. Suppliers only need to ask the user a few questions. This works well if you can reuse the stacks. You can automate to run emulation at scale – not hard for someone within the community to create scripts.

Rechert pointed out the importance of keeping the data separate from the image. A very complex setup may need full system preservation, which means to preserve the full virtual machine with a disk image. One needs to think about dependencies. In this case, his company may also be able to rebuild a machine from a file system or backup.

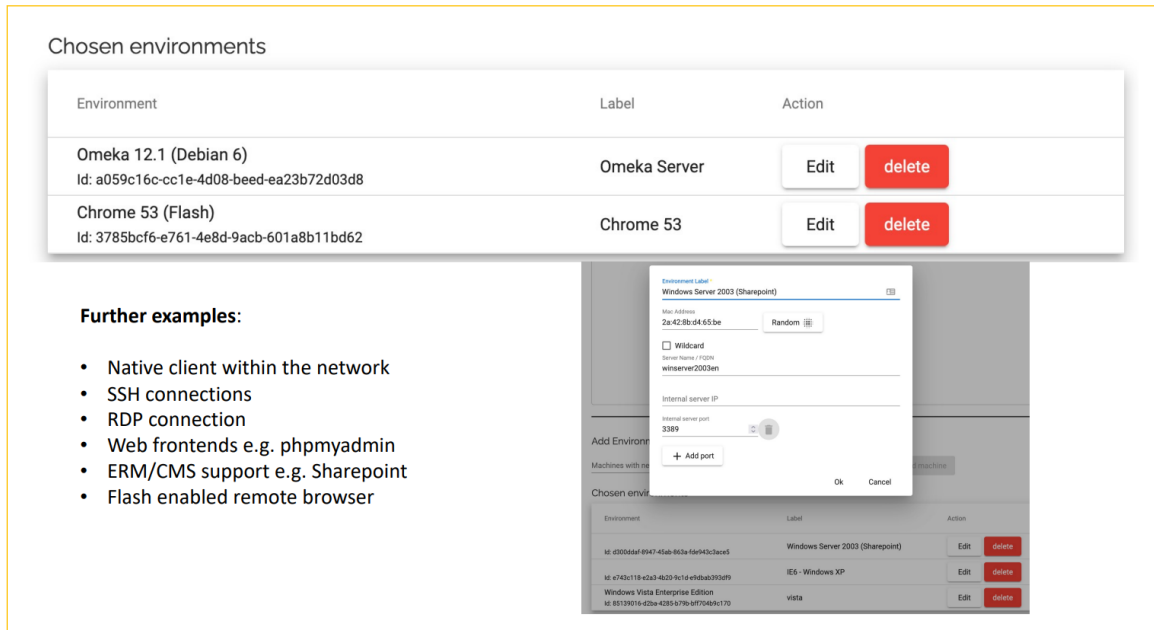


Figure 13: Options for emulated network environments with Stabilize.

He saw several options of how to preserve the software stack. If you follow an emulation strategy for preservation, you must prepare for obsolescence. The good thing is that everyone has the same problem so it is a safe bet that we will see other emulators in the future. A key issue is to preserve the images. Another important observation is around complexity. The complexity and risk profile should guide any option that you take, whether a template-based or full system emulation is deployed. Emulation scales well. The more users, the more people are available to cooperate with (see EaaSI in references).

Emulation is also useful as an access technology. Here, Rechert introduced Stabilize (see references). Emulation allows you to prolong the sunset phase of services. Once you have a software stack you can set up an emulated network. Users need to provide IP address and port info as appropriate, and can “fire up” a machine. An example shown was running on an SQL Server.

Emulation is a software-focused strategy – the complexity of the setup and anticipated risks are key to implementation. Emulation is a tool to offer a prolonged (endless) sunset phase of obsolete services. Operating obsolete software systems remains a huge non-technical, especially legal, problem but there are some technical solutions possible.

Questions and discussion

- Is emulating databases running on mainframes feasible – for example COBOL? How to access those mainframes? Where is the extracted data stored? – Rechert thought mainframe technology mostly didn't



need emulation since the data contained was mostly quite simply organised and could be best preserved via format migration.

- Will Oracle or their successors care for users of their obsolete DBMS versions in 60 years' time? Will they allow users to use their intellectual property at all? – Rechert said this shouldn't be too much of a problem if companies realise that they have no market to sell that product anymore.
- Emulating the whole database is only available for a very small subset of database engines.
- How do you even connect to the old database? New official database drivers refuse to connect to the old versions. The same holds true for ODBC (32-bit version only, for example).
- How do you emulate specific versions of mainframe DB2, for example? Or Oracle? Or Amazon Web Services (AWS) DynamoDB?
- Will someone have data in the same virtual machine? What if there will be a cluster?
- What about the licensing? What about the cost? What if the price models change?
- Databases vary wildly even between versions of the same databases (for example hashing functions change in MySQL).
- Rechert was also uncertain about the survival rate of concrete DBMS versions – people must prepare for obsolescence. If emulation solutions will be around, it would solve the problem for the next 10-20 years. Afterwards, there need to be people who will build the stack to wrap this into a new framework. This would have to be done every so many years.
- One conclusion was that emulation is a very interesting idea, but often hard to implement.

Emulation

- Use Emulation as Access Technology
 - To contextualize the software stack
 - Provide interfaces and interaction to enable data access
 - Rebuild administrative / business processes
 - Prolonged sunset phase of obsolete services – continuous access



Figure 14: Emulation as contextualising and prolonging technology