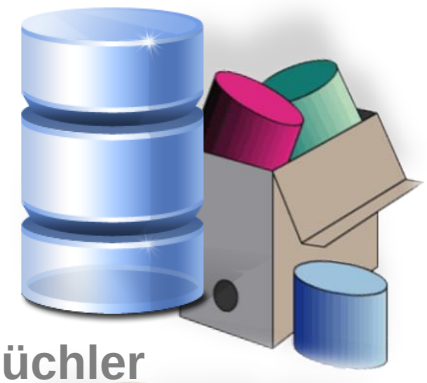
Schweizerische Eidgenossenschaft
Confédération suisse
Confederazione Svizzera
Confederaziun svizra

Eidgenössisches Departement des Innern EDI
Schweizerisches Bundesarchiv BAR
Az:011.2-2015

The benefits of database preservation using SIARD Format Version 2.0



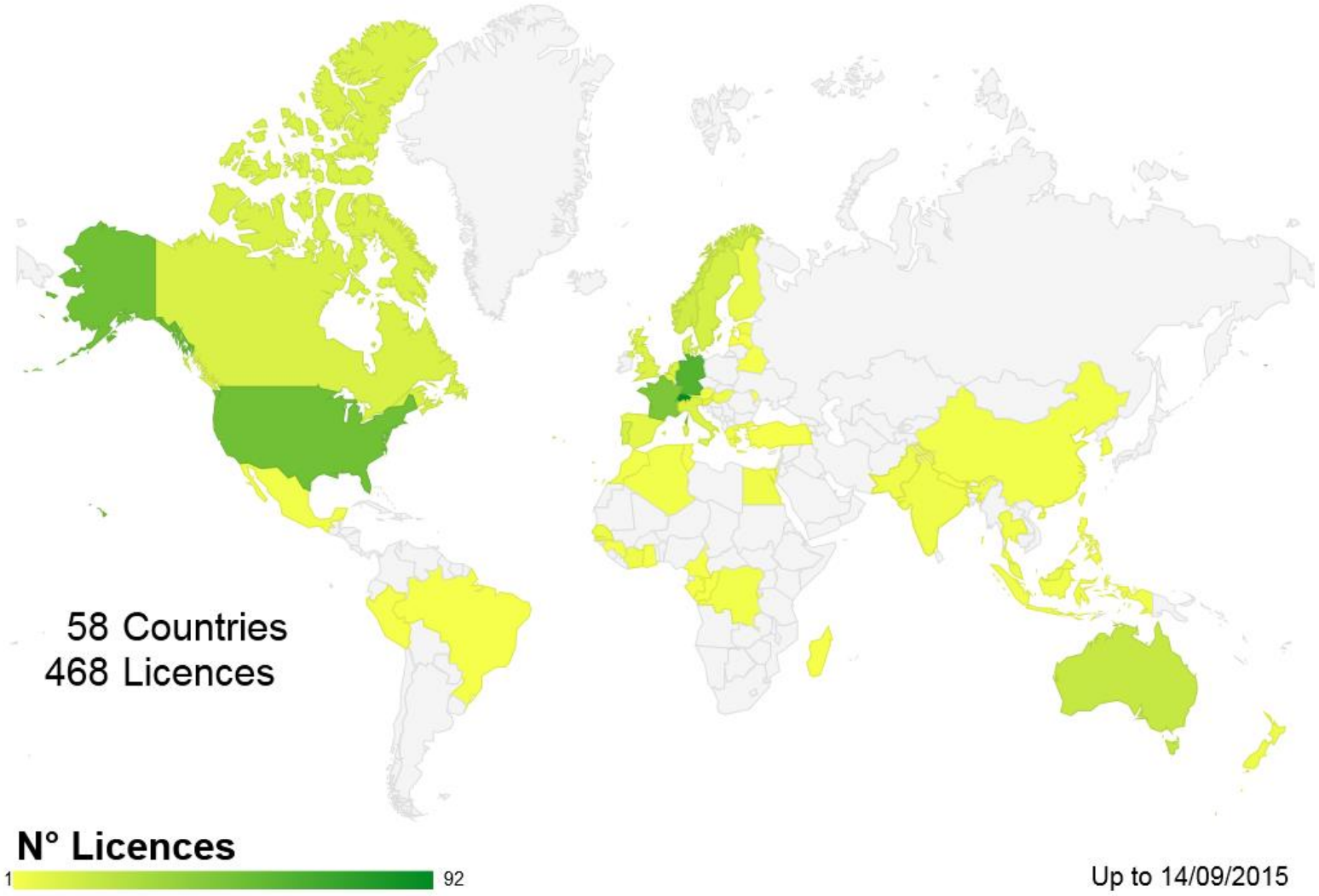
Dr. Krystyna Ohnesorge, Tobias Mérinat, Marcel Büchler
Swiss Federal Archives, SFA

What does SIARD standard for?

Software Independent Archiving of Relational Databases

- **Nature of data:** databases
- **Type:** relational databases
- **Convert content** into archivable format
- **Detachment** of data from executable applications

Who is using SIARD today?



What is a «relational database»?

A database is meant to store data

- efficiently
- free of redundancy and contradiction
- permanently

Why is it called «relational»?

- Data is stored in tables consisting of rows and columns
- Rows indicate data items, columns their characteristics (just like in Excel)
- There are two groups of tables:
  - Entities (the «things») and
  - Relations (how the things relate to each other)

An example of a relational DB

People

- Ueli
- Housi
- Jüre
- Heidi
- Annekäthi

Outdoor sport

- Mountainbiking
- Paragliding
- Mountaineering
- Skiing
- Climbing

Who likes to do what?

- Ueli likes Mountainbiking
- Housi likes Paragliding
- Jüre is a Mountaineer
- Heidi likes both Mountainbiking and Mountaineering
- Annekäthi likes to Climb

… And of course everybody likes to ski in winter!

How can we model these relations?

We could use one table to store all information:

Person	Sports
Ueli	Mountainbiking
Ueli	Skiing
Housi	Paragliding
Housi	Skiing
Jüre	Mountaineering
Jüre	Skiing
Heidi	Mountainbiking
Heidi	Mountaineering
Heidi	Skiing
Annekäthi	Climbing
Annekäthi	Skiing

How can we model these relations?

Challenges with the one-table approach:

- Relations can not be quickly grasped
- It’s not easy to add new sports
- And what if, suddenly, skiing is a thing of the past and everybody gets out their snowboards?
- The chance of introducing inconsistencies is high

Relational model to the rescue!

We use three tables (two entities and one relation) to store all the information efficiently:

Person (entity)

Ueli
Housi
Jüre
Heidi
Annekäthi

Relation «likes»

Ueli	Mountainbiking
Ueli	Skiing
Housi	Paragliding
Housi	Skiing
Jüre	Mountaineering
…	

Sport (entity)

Mountainbiking
Paragliding
Mountaineering
Skiing
Climbing

This representation gives us efficient search in both ways. Now it's easy to add new people and new sports, and those who don't like sports at all are not left out.

Why is that difficult to archive?

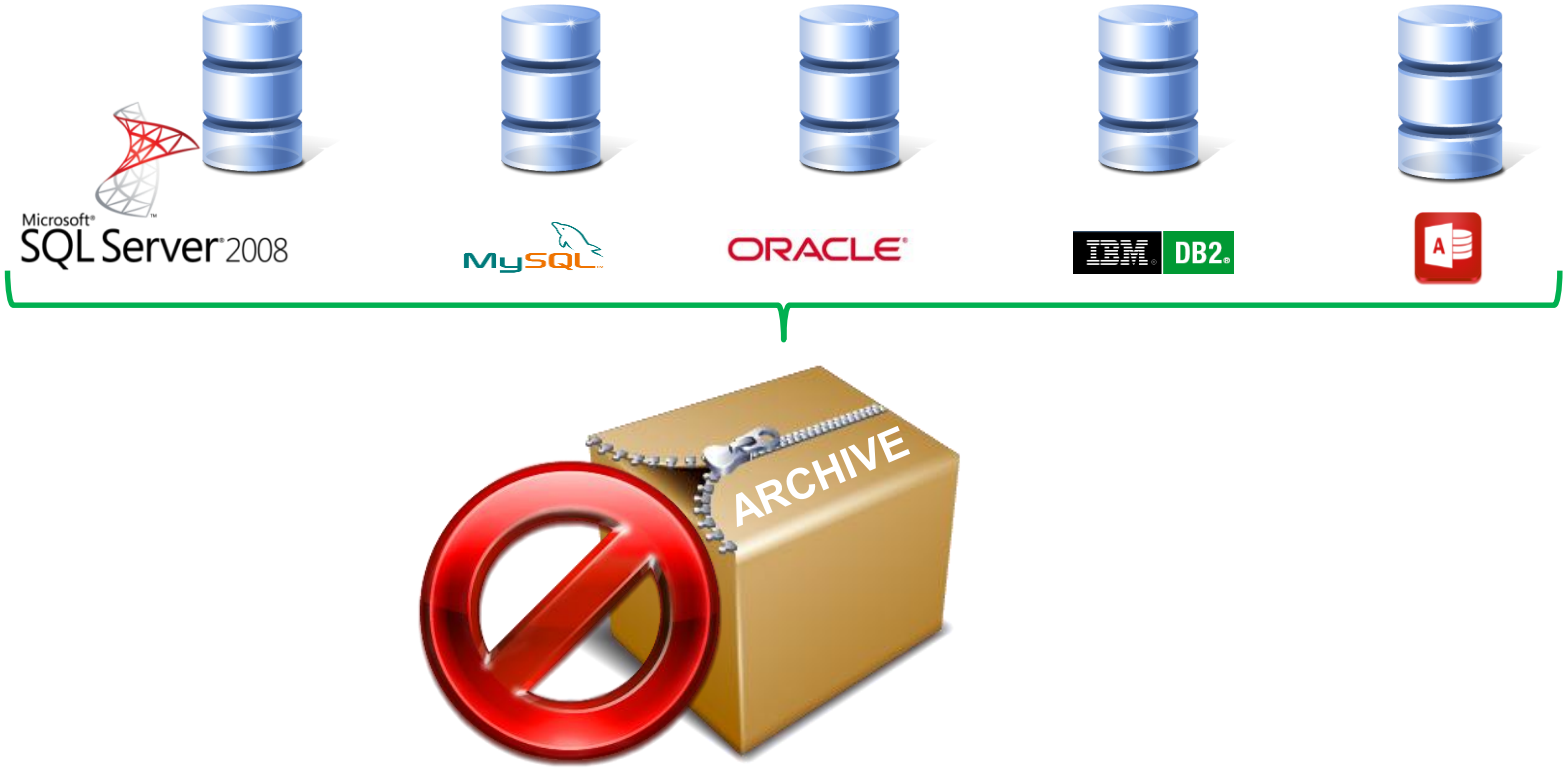
There exist several major database management systems, their storage formats are not compatible.
… And these systems and storage formats evolve over time!

Conclusions:

- we cannot archive proprietary storage formats
- todays storage files are almost certainly

  not readable with tomorrows applications

Proprietary databases are not suited for long-term preservation!



The SFA developed the format SIARD for archiving the contents of relational databases

Database normalisation using SIARD

SIARD Suite is the reference implementation

- (Excel)
- **Microsoft Access**
- **Oracle**
- **Microsoft SQL Server**
- **MySQL**
- **IBM DB2**
- Sybase
- Postgresql
- SQLite

SIARD Suite

SIARD principles

- **Preserve Information**, not layout or interaction
- **Preserve primary data,** not code
- Preserve tables with their **relations**

Functionality Preservation

Constraints

Archived databases are consistent when they are created from consistent databases. Since, once archived, they will not be changed anymore, preserving constraints is not mandatory.

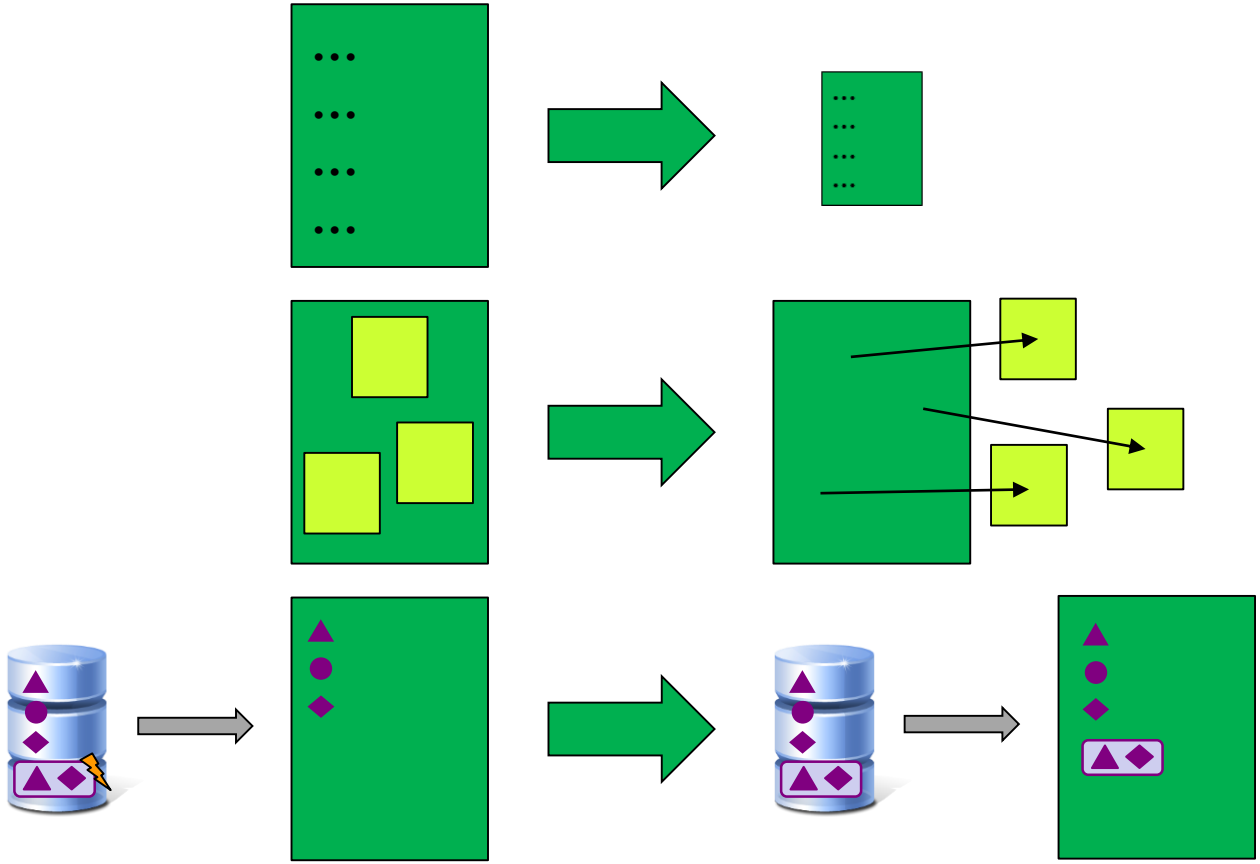
SIARD Format Version 1.0

- The SIARD format saves database-content in **a SIARD-file (SIARD-archive)**
- A SIARD-file is an uncompressed **ZIP-folder (ZIP64)** which contains several XML-files
- There is a single **XML-file** that documents all metadata for the database content, based on **SQL:1999**
- The remaining **XML-files** contain data from the tables (the actual database content)
- The format SIARD is based on **open standards:** SQL:1999, XML, XML Schema, UNICODE

New challenges

- Databases can get huge, which means the SIARD XML Files get huge too
- Everything that lived in the database is stuffed into XML-files, this makes control and preservation harder than necessary
- SIARD is based on SQL:1999, which, by IT standards, one could call ancient

Our solutions



SIARD 2.0 – Changes

- Upgrade of SQL:1999 to **SQL:2008**

- Support for:
  - all SQL:2008 types, in particular **user-defined data types** (UDTs)
  - **binary or character large objects** (BLOBs and CLOBs) outside of the SIARD file using “file:” URIs
  - **deflate** as a compression mechanism

SIARD 2.0 – Benefits

- BLOBs or CLOBs can be stored outside of the XML as self-contained files. They can thus be indexed and accessed directly and more easily migrated to another format. It would also be easier to mandate specific formats (given a policy that requires external storage of binary data).

- Deflate compression minimises (expensive) storage usage.

- With user-defined data types modern databases are supported as they are; no data type conversions are needed when archiving.

Make sure you're not missing on this map!