CREATİNG A SHİNY DASHBOARD FOR A LEGACY İNTEGRATED LİBRARY SYSTEM

MATTİ LASSİLA | JYVÄSKYLÄ UNİVERSİTY LİBRARY | FİNLAND

THE PROBLEM

The integrated library system (ILS) has traditionally been the backbone of all library operations, including the acquisition of resources, cataloguing and collection management. Therefore, a wealth of information is being stored to the ILS and is potentially available for analysis.

Unfortunately, the built-in reporting capabilities of the ILS are usually very limited. Sometimes, these limitations can be circumvented by using an external database query tool if the ILS vendor permits direct SQL-access to the relational database powering the ILS.

THE FORMER SOLUTİON

Previously, we have been using **MS Access** as a primary reporting interface to the **Oracle 11g** database of the ILS. At the request of non-technical staff members, the systems librarian created MS Access queries on ad-hoc basis (Figure 1). This workflow was time-consuming and because of the manual nature of the process, it was impossible to utilize real time information such as book hold statuses or transaction logs.

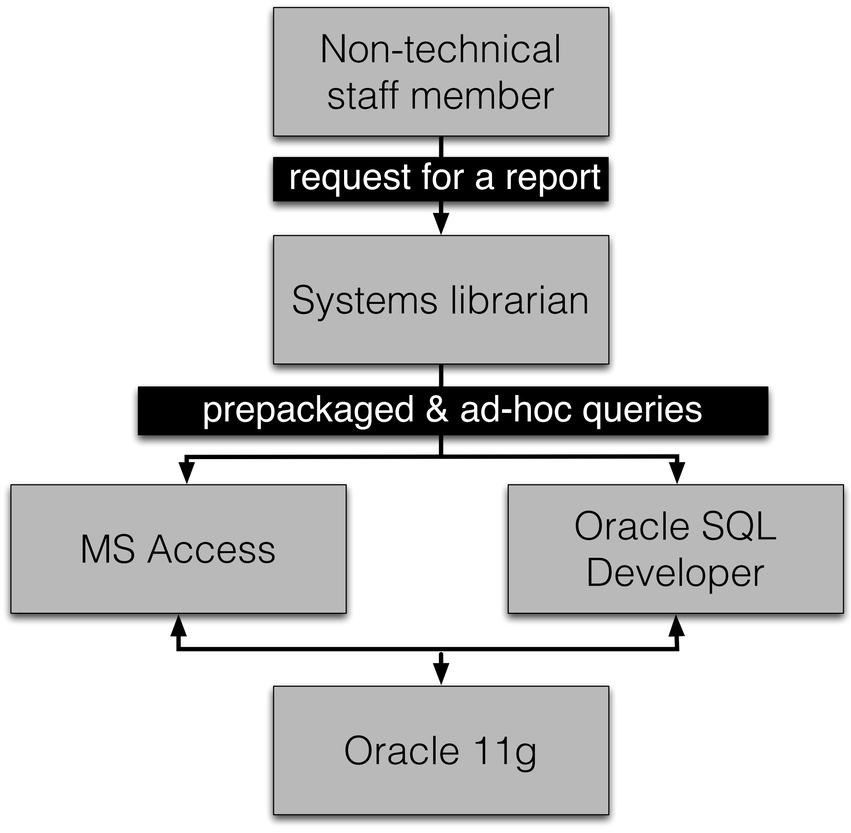

Figure 1. Reporting workflow before the implementation of the dashboard

THE NEW SOLUTİON

Using existing SQL queries as a starting point, we have created a Shiny web app to automatize our reporting process (Figure 2). In addition to Shiny, the key building blocks have been **ROracle** and **scheduleR**, which we are using as a lightweight Extract-Transform-Load (ETL) tool.

For reports requiring real time information, the ILS database is queried directly using ROracle. For database intensive reports, such as visualizations of the collection structure (Figure 3), data is prefetched using scheduleR R scripts, converted to **data.tables** and saved to the disk as **RData** files. Additionally, we are using periodically running R scripts to extract full content from selected database tables for building time series datasets.

External systems, such as the shelf and collection location database, are integrated to the dashboard using **BaseX** XML database as a middleware. BaseX allows easy combining of heterogeneous data sources using industry-standard **XQuery** language. Interfacing to the dashboard is done via BaseX **REST API**.

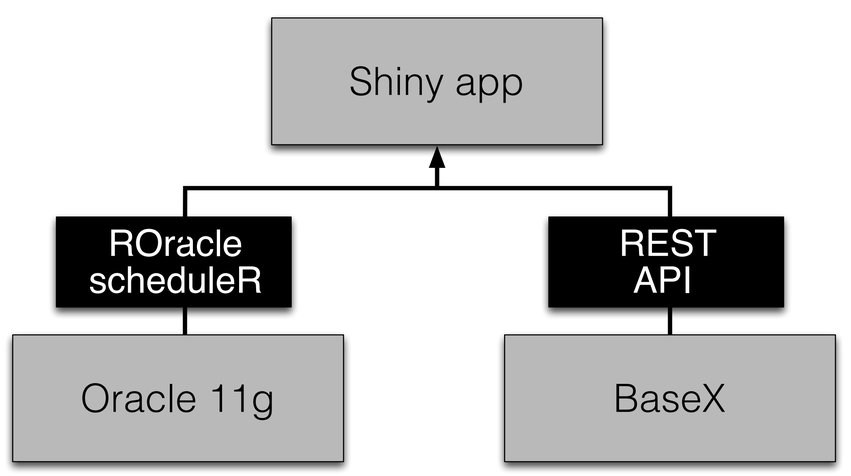

Figure 2. Components of the dashboard

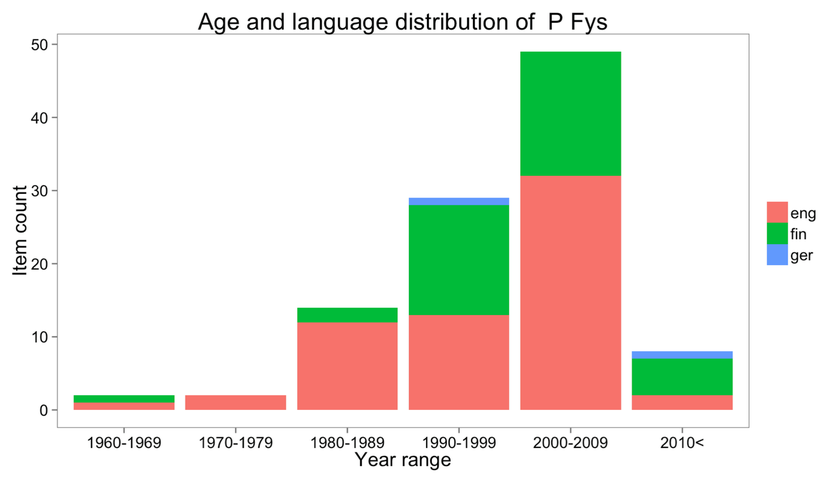

Figure 3. Example of collection visualization