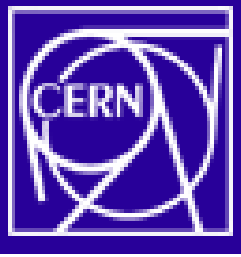
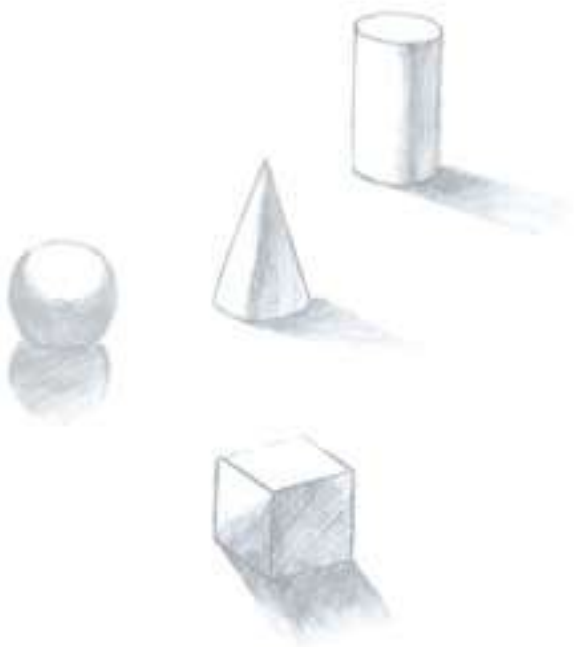


Database Lookup Service

Kuba Zajączkowski
Chi-Wei Wang

Outline

- Database Service Lookup basics
- Database Catalog
- Choosing a best suited database
- Current status
- Future possibilities



Database Lookup Service

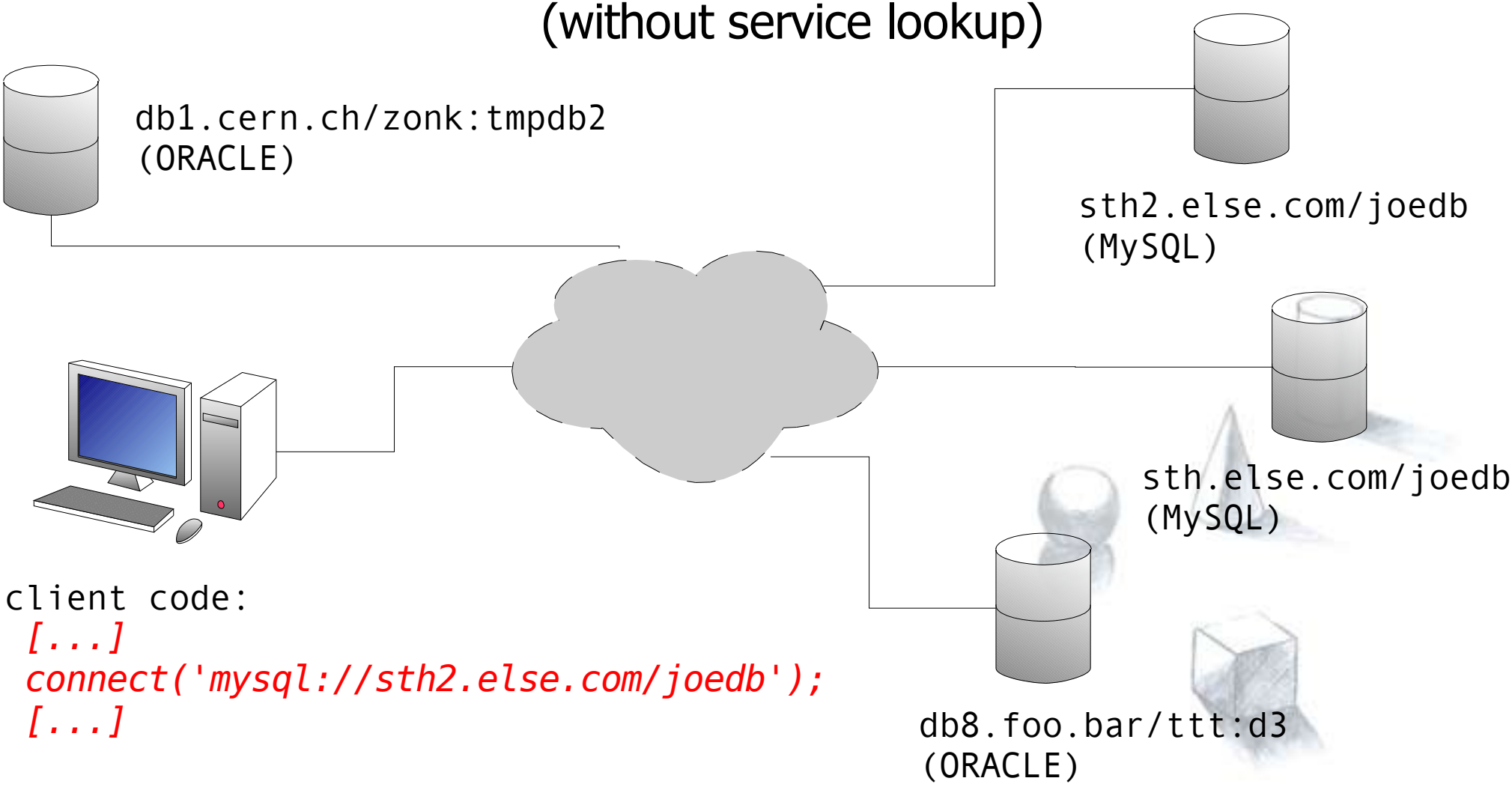
Problem Statement

- Current applications use physical connection strings
- In grid environment connection strings have to be obtained from a catalog so that job is location independent

Database Service Lookup

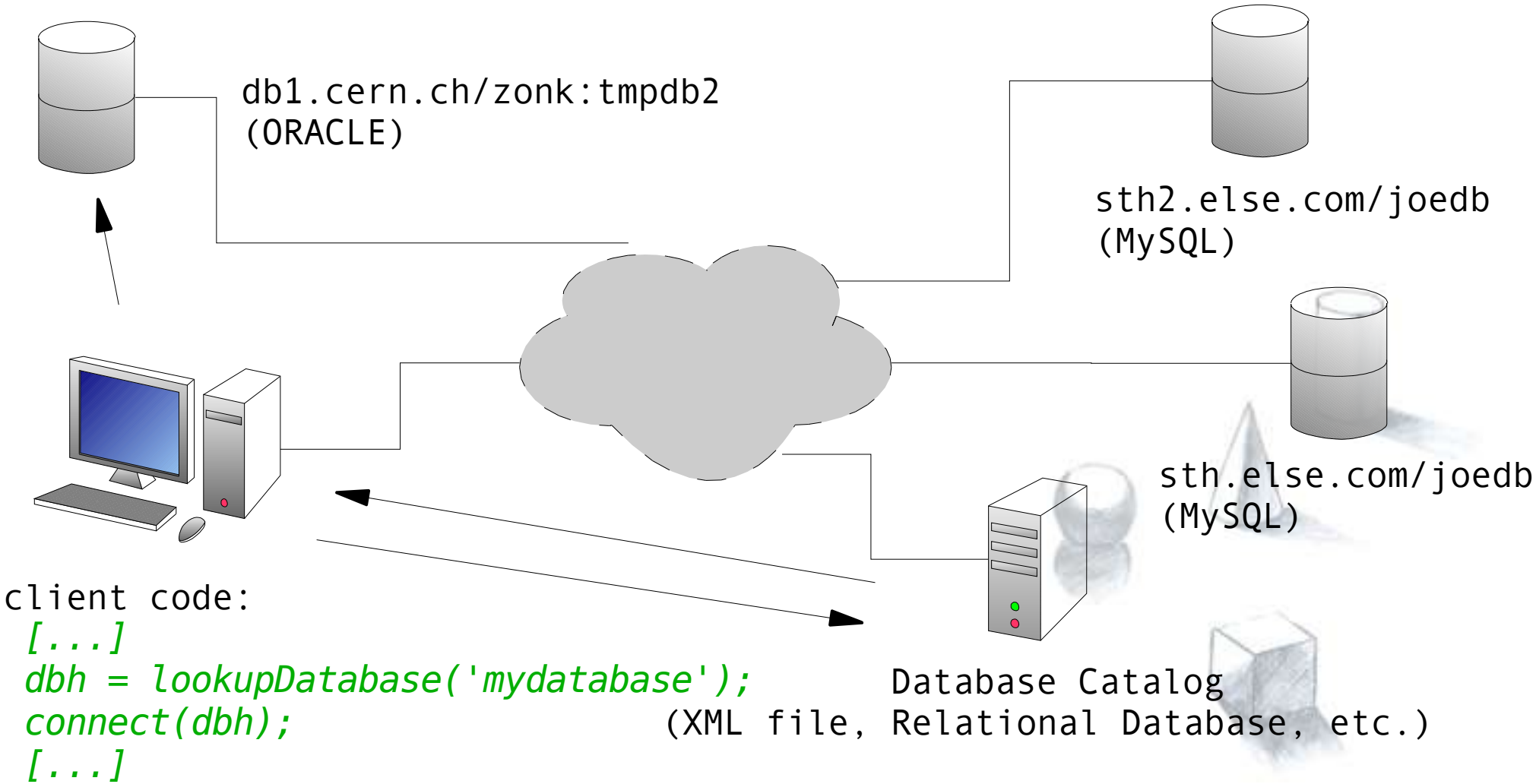
Usual case

(without service lookup)



Database Service Lookup

With Service Lookup



Database Service Lookup

- Pros
  - No need to know the exact connection strings
  - The best database chosen on runtime
  - Includes a simple database failover mechanism
  - End user doesn't need to follow physical database location changes
- Cons
  - Additional step before a real connection
  - The Database Catalog needs to be maintained

Database Catalog

- Terms
  - Logical Database
  - Physical Database (connection string)
  - GUID (Globally Unique IDentifier)
- Schema

- No usernames or passwords stored!

Database Catalog Functionality

- Basic Functionality
  - Lookup all physical database connection strings with a given logical database name
  - Find a good (possibly the best) database connection available
  - API to manage the catalog (register new, add, modify, delete, etc.)
- Additional Functionality
  - Query and management tools (command-line, etc.)
  - Queries done on multiple catalogs

Choosing a best database

It's important to select a best physical database service available. The best service means at least that it's alive and it's the fastest one to access data at that time.

method	pros, cons
ping server	select a real alive, shortest response time (ping or ..) server
	unknown server load, adds some (little) load to server
suffix rule (.ch,.tw,...)	select a server with theoretically shortest response time, doesn't affect the server
	unknown server load, doesn't consider real situaton
daemon on every server	detailed information of service status, server load, network status...
	inconvenient to install and maintain, add some load
small real test	trying real case on every server, we can determine the best one
	adds significant load on servers

Current Status

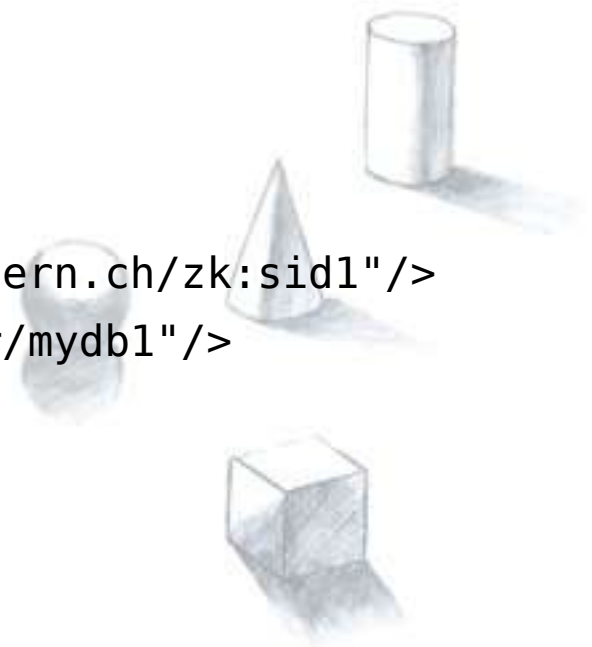
- First prototype implementation of catalog stucture and tools is ready (alpha!)
- Can be easily build as a POOL module, using the same infrastructure
- Exact requirements are still being defined (metadata, configuration options, avaiability?)
- Now heavily based on POOL File Catalog (reusage of the same code, less work required)
- Database choosing procedures in development (ping prototype ready), as of now not merged with the rest of the catalog

Working Prototypes

- XMLDBCatalog
  - Catalog stored in xml file
  - Access over local file system or http (readonly)
  - Now uses POOLFileCatalog DTD, future versions will use it's own or more generic DTD

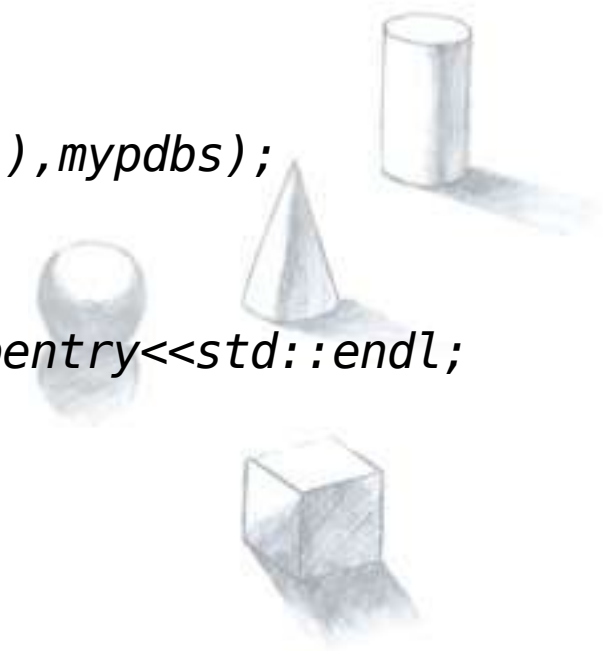
Example:

```
<File ID="ECB56706-4449-D911-9B29-0007E971D10B">
   <physical>
      <pfn filetype="oracle" name="oracle://db1.cern.ch/zk:sid1"/>
      <pfn filetype="mysql" name="mysql://foo.bar/mydb1"/>
   </physical>
   <logical>
      <lfn filetype="" name="mydatabase1"/>
    </logical>
</File>
```



Example Lookup

```
IDatabaseCatalog* mycatalog;
DClookup l;
mycatalog=new IDatabaseCatalog;
mycatalog->setCatalog("xmlcatalog_file:dbcatalog.xml");
mycatalog->setAction(l);
mycatalog->connect();
mycatalog->start();
PDBContainer mypdbs(mycatalog);
mypdbs.reset();
l.lookupPDBByLDB(std::string("logicaldb"),mypdbs);
while(mypdbs.hasNext()){
PDBEntry pentry=mypdbs.Next();
    std::cout<<"connnection string: "<<pentry<<std::endl;
}
mycatalog->commit();
mycatalog->disconnect();
if(mycatalog=!0) delete mycatalog;
```



Short term tasks

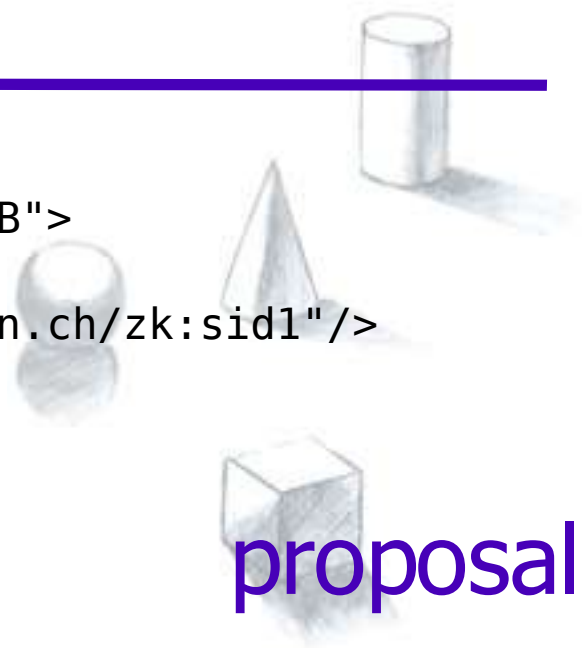
- Simplified interface for basic query (physical database connection string lookup based on logical name)
- Finalization of requirements and design
- Integration with Chi-Wei's choosing functions
- Final definition of a DTD for XML (wheather it should be specific for Database Catalog or more generic)

XML Catalog DTD

```
<POOLFILECATALOG>
<File ID="ECB56706-4449-D911-9B29-0007E971D10B">
   <physical>
      <pfn filetype="oracle" name="oracle://db1.cern.ch/zk:sid1"/>
   </physical>
   <logical>
      <lfn filetype="" name="mydatabase1"/>
    </logical>
</File>
</POOLFILECATALOG>
```

current

```
<POOLCATALOG type="DATABASE">
<Entry ID="ECB56706-4449-D911-9B29-0007E971D10B">
   <physical>
      <pn type="oracle" name="oracle://db1.cern.ch/zk:sid1"/>
   </physical>
   <logical>
      <ln type="" name="mydatabase1"/>
    </logical>
</Entry>
</POOLCATALOG>
```



proposal

Future

- RAL (Relational Abstraction Layer) based implementation (there already is an existing RAL implementation of File Catalog)
- Bootstraping – finding a Database Catalog to use on startup
- Integration with existing POOL Services (using Database Catalog by default)
- Implementation based on Grid middleware? (it could be possible to implement Database Catalog using the grid file catalog, but it may interfere with other software using gfc which assumes only file names there)

Summary

- In distributed environment there is a need for Database Catalog
- First prototype is ready
- Project is based on POOL File Catalog which makes development much easier/faster
- Project is not production-ready, it requires futher developement

Thank You!

For more information
feel free to contact us:

Kuba Zajaczkowski
<Kuba.Zajaczkowski@cern.ch>

Chi-Wei Wang
<Chi-Wei.Wang@cern.ch>