Reworking Deep-sky catalogs & retrieval of data on unknown objects in KStars from the internet

Christian Dersch

March 24, 2016

1 Introduction

KStars is a full-featured desktop planetarium and provides a set of star catalogs (e.g. USNO NOMAD) and deep-sky object (DSO) catalogs (e.g. Steinicke NGC/IC) to access a huge amount of objects. Many objects are included in several catalogs, this leads to a set of improvements that can be done for KStars.

Due to the large number of stars and deep-sky objects, a complete catalog would be an extremely large database (several Gigabytes). Instead of using a local copy of such a database it is possible to provide the data online as a web service. This is a quite common technique in the astronomical field resulting in the *Virtual Observatory*[1] (VO) initiative. One of the providers for VO-related services is the Strasbourg astronomical Data Center[2] (CDS). To provide information about additional objects in KStars it would be a nice feature to make use of the services provided by CDS, especially the *Sesame Name Resolver*. This service queries the SIMBAD database providing access to several catalogs.

The second issue is the cross identification of objects which are listed in more than one catalog. This is a very common case for astronomical objects as almost all surveys produce their own catalogs. In the case of KStars the provided DSO catalogs are affected, some objects are listed in two or more catalogs without markers for cross identification.

2 Project goals

2.1 Enhance KStars to be able to query CDS Sesame

The main idea of this topic is now to enhance the KStars search for objects by an option to query Sesame and store the result in the users local object database. There are two possible outcomes of the local database query:

1. the queried object is known → no further action required
2. the object is unknown → we will query Sesame now
   (a) object is known and just an alias for an object (e.g. WR134 for HD191765) known in local database
   (b) object is known and not another identifier for a known object
   (c) object unknown → return message that object is unknown

While case (2.b) is the most obvious one, (2.a) is a quite common situation too. Sesame optionally provides a list of aliases of the searched object. By using this additional information the user is more flexible, he can use the alias and does not have to search for the number in Henry Draper catalog first. For (2.c) a dialog for manual input of position, magnitude etc. could be provided.

2.2 Rework DSO catalogs to support cross identifications

For deep sky objects the catalogs provided by KStars are affected by missing cross identification, e.g. NGC 6742 alias Apell 50 is listed in both abell_pn.cat (Apell 50) and ngcic.dat (NGC 6742). As a result two markers are drawn in the sky map instead of only one. The goal is a rework of the DSO catalogs to adress these issues and develop a structure based on a relational database (SQL) which also ships information on other designations. The advantage of SQL over the current implementation based on ASCII tables is a higher flexibility. Also SQL allows to perform database joins when adding new objects which are just another designtation for an already existing object. This solves the problem of missing cross identification.

[1] `http://www.ivoa.net/`
[2] `http://cds.u-strasbg.fr/`

2.3 Custom database for objects not in local catalogs

To improve the usability of the search it is very useful to save already queried objects in a custom database, it is a quite common case that a user does not search for a specific object only once. Even more important is the possibility to work offline in the outback for observations. With a custom database the user is able to add the interesting objects prior to observations and make use of Sesame even if offline at observation time. Alongside with the implementation of the custom database a possibility for manipulation of the same has to be provided to manage existing objects and manually add new ones. This goal is correlated with the rework of DSO catalogs as simiar database structures will be used.

2.4 Nice to have: Query other CDS services too

It would be a nice feature to be able to query other web services too. As a starting point a general concept for querying remote databases will be presented in section about implementation.

2.5 Nice to have: Alias database

While the problem of aliases/designations for DSO will be addressed with the DSO catalog rework, there is no solution yet for stars. Star catalogs are handled completely different in KStars for performance and resource usage reasons. A starting point could be an alias support for stars shipped with an identifier (e.g. Henry Draper index). Unfortunately most stars (i.e. all stars in USNO NOMAD) are shipped as a binary file without identifiers. In this case a remote query to SIMBAD using coordinates could be an approach, that requires the implementation of the previous "Nice to have".

3 Implementation

The implementation consists of four major parts:

1. a class and/or small library to query Sesame resolver
2. rework DSO catalogs
3. custom database to store objects
4. integration into KStars GUI

3.1 General aspects of the implementation

Developing and enhancing code for a mature project like KStars specifies the requirements of the implementaion quite detailed. The new code has to match the existing structures (e.g. class structures, databases) to integrate well. Beside these obvious requirements it makes also sense to have a look at other planned features to avoid conflicts. By looking at the list of GSoC 2016 ideas we get an impression about planned features:

1. KStars Lite
2. KStars on Windows
3. DSS Overlay

While the DSS overlay is independent from this project, the projects "KStars Lite" and "KStars on Windows" have to be taken into account. KStars Lite is a project to bring a basic KStars edition to tablets and mobile devices. On such devices it is not easy to introduce additional dependencies in general, so a small dependency footprint is required. For the KStars on Windows it is required to have all additional dependencies availabe for Windows too. Both features require a cross-platform capable implementation of our project.

Thankfully the Qt application framework used by KDE and so KStars provides classes to perform HTTP requests, access several types of SQL databases (for KStars: SQLite 3.x), parse XML files and is a cross-platform framework available on many platforms including the established mobile platforms like Android and iOS. So by limiting on Qt as a dependeny and avoiding any platform specific code, which should be possible for this project, there will be no conflict with further developments of KStars.

3.2 Class/Library to query CDS

The Sesame resolver provided by CDS provides a simple HTTP-GET service[3] to query for objects. To perform the request the QNetworkAccessManager, QNetworkRequest and QNetworkReply classes provided by Qt Network will be used. Sesame provides options to get plain text or XML output, as XML is a well defined standard implemented by Qt XML classes, XML will be chosen to extract the interesting values (ra, dec, mag etc.) from the request.

The implementation scheme:

- an abstract class QueryRemoteService which defines a general set methods required for querying objects
- an inherited class QuerySesameService which implements the logic specific for Sesame requests, i.e. XML parsing

It makes sense to start with an abstract class QueryRemoteService as there are many other services which could be interesting for KStars too (i.e. SIMBAD services) and we can use one abstract logic for inclusion in KStars. To extend to other services one just has to implement an inherited class QueryAnotherService (and add it to a small selection dialog) to get it supported by KStars.
As these classes are quite generic and only rely on Qt, it is possible to provide it as a seperate library to make it available for other projects too.

3.3 Rework DSO catalog handling

DSO are handled by class DeepSkyComponent. Currently the deep sky catalogs are shipped as ASCII tables which are loaded my the method DeepSkyComponent::loadData(). This creates a set of indices and lists of DSO from the available catalogs:

- general deep sky
- Messier
- NGC
- IC
- other

The class DeepSkyComponent also provides the capabilies to draw the objects and search for objects (implemented using hash tables). The objects by themselves are represented by the class DeepSkyObject. This class enhances SkyObject (which holds general values for objects, as type, positions, magnitude) for DSO specific properties, e.g. minor axis and DSO catalog identifiers.

To address issues with cross identification (also with additional custom objects), the ASCII tables should be replaced by SQLite databases. Special fields for other designations will ensure an easy capability for cross identifications, for example using SQL joins. This change requires a rewrite of class DeepSkyComponent, for now I'm quite sure that this is possible without breaking the API there. The method draw() responsible for drawing the DSO will be modified to draw them only once also when they are listed in multiple catalogs. To sum up, the following (re-) implementations and tasks have to be done:

1. create a new SQLite databases for DSO to replace the current ASCII tables and add fields for other designations
2. reimplement class DeepSkyComponent for SQL
3. implement cross identification in DeepSkyComponent using SQL joins (using designation) and thus avoid having one object multiple times in the object list to be used for drawing etc.

For implementation SQLite will be used through Qt SQL classes providing a driver for SQLite. This is already used in KStars for other databases.

[3] `http://cdsweb.u-strasbg.fr/doc/sesame.htx`

3.4 Custom database

KStars already provides a set of classes to describe sky objects: SkyComponent. For the custom database a new subclass of SkyComponent has to be derived. This structure allows a fine integration into existing KStars structure. The database itself is a SQLite database similar to the new DSO implementation, skycomponents.sqlite and other local databases used by KStars like userdb. Using the same database structure as in DSO rework allows performing SQL joins here too. So if an object is added after querying CDS Sesame (which also provides other designations we store in database), the custom database will integrate seamless with DSO database and allow cross identification through SQL joins too.

3.5 Nice to have: Alias database

Right now the issues of other designations and cross identification were addressed for deep sky objects. As this issue also affects stars, it would be nice to have a solution there too. While migrating the entire star catalogs from the current binary format to SQL databases is not a viable solution it is at least possible to create an alias database which connects the HD index with other designations queried using Sesame. For non-HD stars the cross-identification could be done using a coordinate based match using the function DeepStarComponent::objectNearest() with a small search radius and a comparison of magnitudes to avoid wrong identifications.

3.6 Integration into KStars GUI

Finally the capability to query objects online and store them in the custom database has to be integrated into KStars GUI. Thereforce several modifications are required:

1. object search has to be modified for remote query, some rework is suggested as the current search provides suggestions when typing an object identifier. May be misleading as this does not work with online queries for specific objects
2. dialogs, documentation and other strings have to be modified, i.e. dialog which asks user whether new object should be added to local database or not
3. option to show objects in custom database in skymap
4. add a wizard to manipulate the custom database (delete objects, manually add new objects)
5. add configuration options in KStars preferences (user should be able to disable online features, option to add new objects to database in general)

4 Timeline

The projects description of the implementation already shows a set of different targets. This leads to the following timeline:

- **Upto May 23:** Getting to know KDE and especially KStars developers, Studying existing KStars database structures and relevant parts of KStars API. Analyze DSO catalogs to create a matching database layout. Get an overview of other astronomical web services to create a common design for QueryRemoteService class.
- **Target 1 (1 week):** Implement the class QueryRemoteService and the derived class for Sesame resolver QuerySesameService. Decide whether the new classes will be shipped as a seperate library.
- **Target 2 (3 weeks):** Modifiy/reimplement class DeepSkyComponent to use SQL instead of ASCII tables, create SQLite database for DSO by reworking current catalogs.
- **Target 2 (2 weeks):** Implement a class CustomSkyComponent to describe queried objects, create SQLite database for custom objects similar to skycomponents.db
- **Target 3 (2-3 weeks):** Integrate the new search and databases into the KStars GUI
- **Target 4 (1 week):** Modify and enhance end-user documentation for the new features
- **Target 5 (1-2 weeks):** Look at the nice to have (alias database, additional web services). While working with KStars databases and catalogs: Try to figure out whether Skychart catalogs can be usable for KStars too.
- **Clean up week:** Clean up code, fix bugs, final discussion, documentation clean up

5 About me

My name is Christian Dersch, I'm currently studying physics at the Philipps University of Marburg, Germany. My current research topics are astronomy and astrophysics where I use KStars as a software to plan observations and perform measurements thanks to Ekos.

While I'm using KDE for about 13 years now, my main contributions to FOSS are in the field of software distribution. I'm working as an ambassador and packager for Fedora since 2013 and contributing to the Special Interest Groups (SIG) for Astronomy, KDE and Science & Technology. The current project is the Astronomy Spin[4] to be released with Fedora 24. This is a special remix of Fedora for both amateur astronomers and professional scientists. KStars and INDI are the main features for observational astronomy, users get an easy way to start with KStars and observations, thanks to INDI a huge bunch of equipment can be controlled. Besides the contributions to Fedora I'm acting as a supporter and packager for the INDI project.

I have a good knowledge of C++ and Python and I'm already familiar with Qt (starting with Qt4) using both languages (Python via PyQt). I have a coding experience of about 9 years now and already gave two courses/workshops "Introduction into programming with Python for physicists" at Physics Department at Philipps University of Marburg. Due to my work with astronomical databases I'm also familiar with SQL and XML (as VOTables are XML structures).

For now I supported and contributed to KStars in different ways:

- Small fixes, e.g. for issues I found while packaging
  1. Ekos: `https://bugs.kde.org/show_bug.cgi?id=348880`
  2. Namespaces fix (required for GCC 6.x): `https://git.reviewboard.kde.org/r/126976/`
- Supporting other users in INDI/Ekos forum (Nick: lupinix): http://indilib.org/forum.html
- Introduced KStars/Ekos and INDI as a solution for remote controlled observatories, coauthor of publication: "U-SmART : Small Robotic Telescopes for Universities" by Deb et al, Publications of The Korean Astronomical Society, vol. 30, issue 2, pp. 683-685, 2015 `http://adsabs.harvard.edu/abs/2015PKAS...30..683G`

Now I'm looking forward to get deeper involved into KStars. As I don't have any examinations and lectures in summer I'm able to work at KStars full time. I will provide a blog to present my progress and updates and plan to continue contributing to KStars after GSoC as I'm using it for years now and like to improve it in future too. Also I'm only applying for KStars at GSoC 2016.

Contact informtion:

- Email: christian.dersch@gmail.com, at mailing lists: lupinix@mailbox.org
- Jabber: lupinix@jabber.de
- IRC: lupinix at Freenode

[4] `https://fedoraproject.org/wiki/Changes/Astronomy_Spin`