PropBase “Warehouse” Architecture

PropBase is a “data warehouse” system that extracts, transforms and loads data into a simplified data model from across BGS’s heterogeneous property data sources into a single view so that the data is compatible and accessible from a single interface. The system consists of: data tables that form the core of a simplified data structure; coding routines that are run at regular intervals for the extraction, transformation and load of data into the simplified data structures, a second tier partitioned denormalized data access layer that serves as the data access point by applications.

The system also includes a suite of java coded search utilities that facilitate easy data discovery and download to allow for the complex synthesis of many data types simultaneously. ; There is also a web service to allow for machine-to-machine interaction, enabling other software systems to directly interrogate the datasets to visualise and manipulate them.

This system will have a significant impact by allowing multiple datasets to be rapidly integrated for scientific understanding whilst ensuring that data is properly managed and available for future use.

The component parts are:

- Core Data Model
- Extraction, transformation and loading routines
- Data Access Layer
- Web services
- Data discovery and download tools

Core Data Model

The core data model is based around the concept of a summary layer to present complex data in a simple but often denormnalized set of tables and other programmable units within a relational database system. The data layer brings together 3D (x, y, z) property information from various databases each with their own relational structure into a generalised structure so that there's a single consistent point of access of the data for any applications that may require the data. The data model is implemented within an oracle relational database system where the source databases also reside or are re-engineered into to facilitate easy loading of the data. The denormalization techniques used are not unique to oracle and can be implemented in other RDBMS.

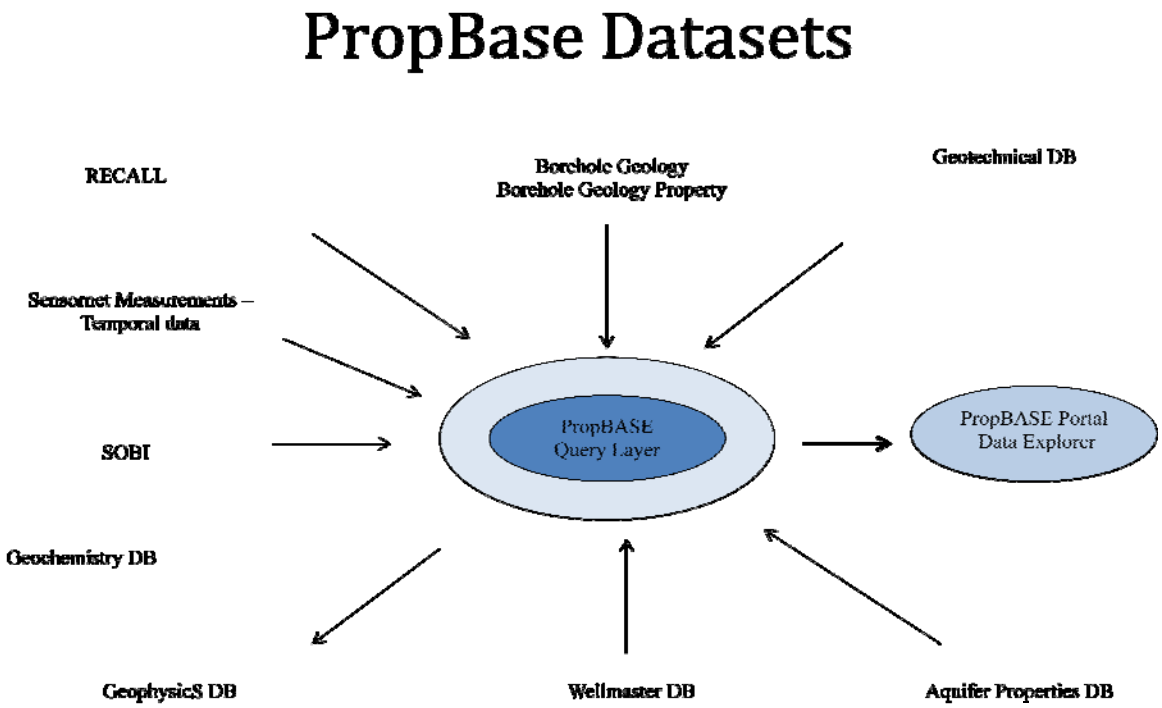
The data structure contains a main table PRB_DATA with the following key attribution:

- a unique identifier
- the data source
- the unique identifier from the parent database for traceability
- the 3D location (x, y, z)
- the property type
- the property value
- the units
- necessary qualifiers
- precision information and an audit trail

Martin Nayembil, Anne Richardson, Graham Smith, Simon Burden, 06/03/2014

The data model also incorporates a series of dictionaries to constrain some of the attributes to include: the data sources, property types, units of measure and co-ordinate referencing systems. The dictionaries in some cases have been collated from the existing terms used in the different databases from which data has been extracted to populate the propbase layer. The property dictionary is a key component of the structure as this defines what properties and inherited hierarchies are to be coded and also guides the process as to what and how these are to extracted from the structure.

The data model is a spatially aware data structure incorporating oracle spatial functionality to represent the various geometries recorded for easy access, integration and visualisation of those geometries in various applications.



Extraction, Transformation and Loading (ETL) Routines

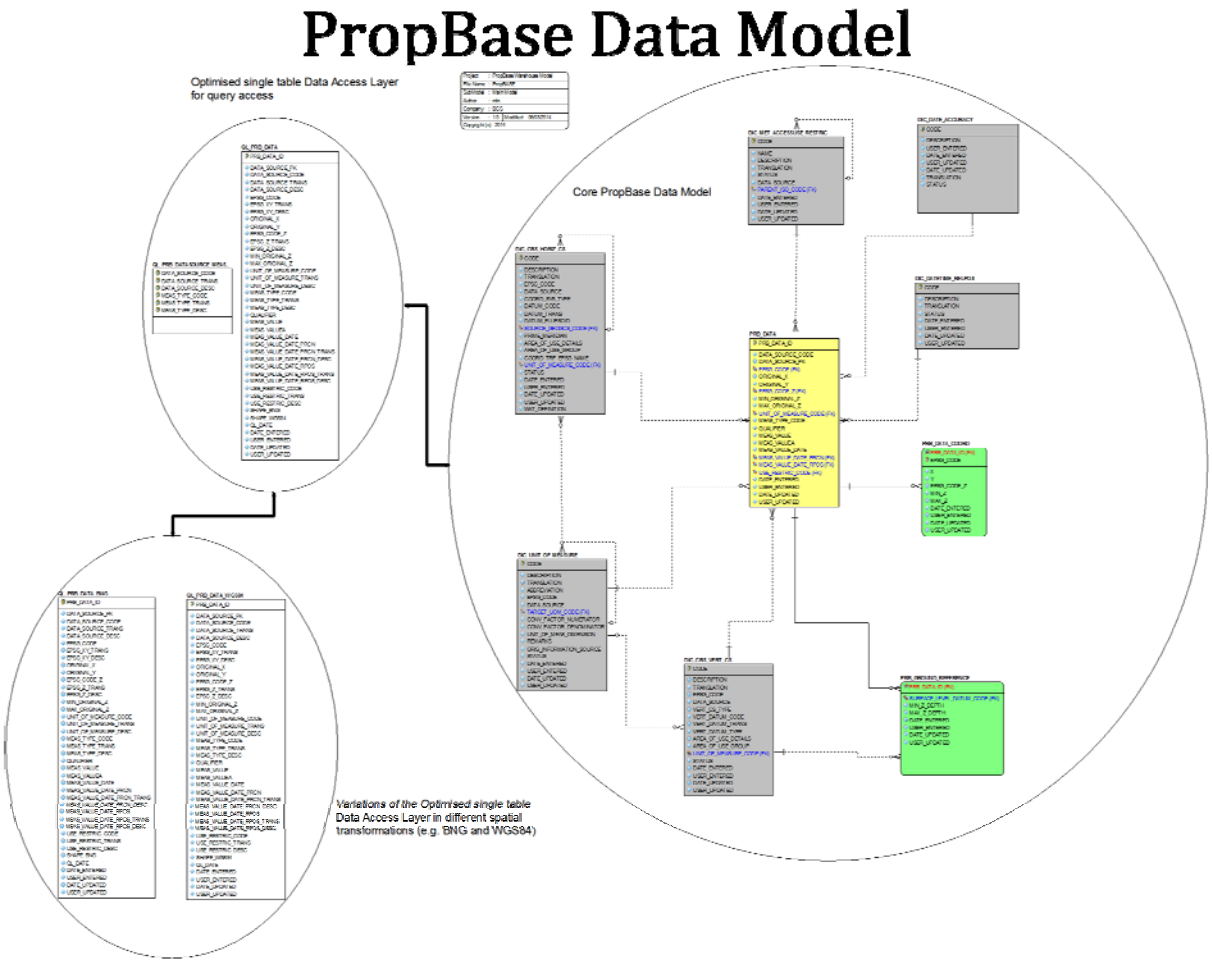
Given the many heterogeneous data sources, property types with different data extraction and transformation requirements, the need for a full audit trail, linkages to source records and regular updates whilst maintaining the richness and integrity of the data, it's important that there's a co-ordinated technical approach to keep the “warehouse” synchronised. The PropBase layer uses a number oracle functions/procedures/packages written in PL/SQL containing the necessary logic to carry out the data extraction, transformation and loading processes (data manipulation) to keep the contents of layer synchronised with those of the underlying databases. These procedures and/or packages are then run as scheduled jobs at regular intervals (e.g. weekly) or can be invoked on demand.

Martin Nayembil, Anne Richardson, Graham Smith, Simon Burden, 06/03/2014

Data Access Layer

The Propbase warehouse architecture includes a highly optimised table as the data access layer for applications and other. The layer is an extra denormalized table on the core propbase data model, optimised as the main point of call for the PropBase Webservice. The tables in the core data model are never accessed directly but only used to collate and harmonise the data from the heterogeneous data sources to populate the data access layer. It's was created by pre-joining the data tables that make up the core data model, with decoded dictionary values and also spatially enabled to provide a single access object for propbase data.

The layer is indexed appropriately and also partitioned to optimise query access by applications and other. Also provides greater flexibility in further optimising, adding or removing data attributes/data and the presentation/extraction of data in different spatial transformations and other.



Webserver Web Service

The Propbase web service component is created in Java using RESTful principles. It is developed using a number of BGS and open source class libraries, the most important in terms of interfacing design is the Restlet library, which provides the HTTP interactions required by the service, including but not limited to HTTP standard content negotiation and internally URL to controller routing and

setting of appropriate HTTP headers, including enabling the setting of sensible cache headers, to allow clients to perform their own caching.

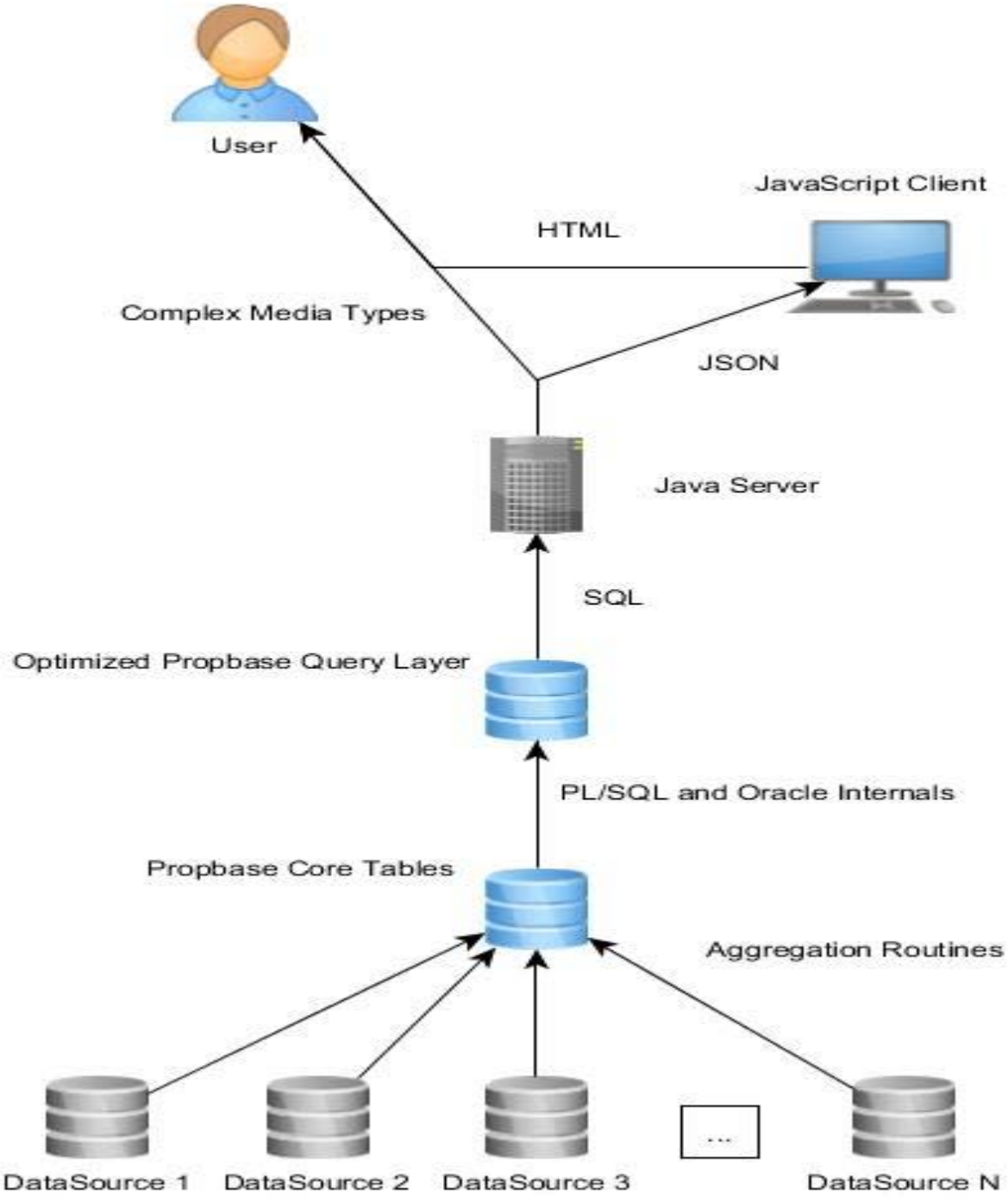
This Application contains interactions with the database using a series of classes and queries that have been optimised alongside the data access objects. These provide a number of caches in memory on the Java server of the various core dictionary objects and various queries against the data warehouse layer.

Server side caches of dictionaries are currently maintained for a little under a day, as the dictionaries are pretty stable it is not envisioned that this latency should cause many issues. Additionally paging operations are set and available via our library on all of the core queries. To provide data to users as fast as possible and to prevent server overload of massive datasets, in addition to the page of data the total available results for each unpaged search are lazily calculated and stored for up to a day to reduce database load through constant recalculation by the Oracle database. The number of threads available to make these lazy requests has been set to avoid using the entirety of the database connection pool, allowing new data requests to continue to be made. Totals are therefore provided to the client if and only if they are already available, to reduce wait times on the data, as users will often request, several pages based upon the same base query. It is envisioned that this should be available in a reasonable number of requests.

Data is made available in a number of different formats to clients. Individual Item data is available in JSON, RDF, KML and HTML formats while lists of items can be obtained in JSON, CSV, TSV, KML, Atom, and Shapefile as both bases and vertical line segments. Information about datatypes and measurement types are only available in JSON and HTML at the moment.

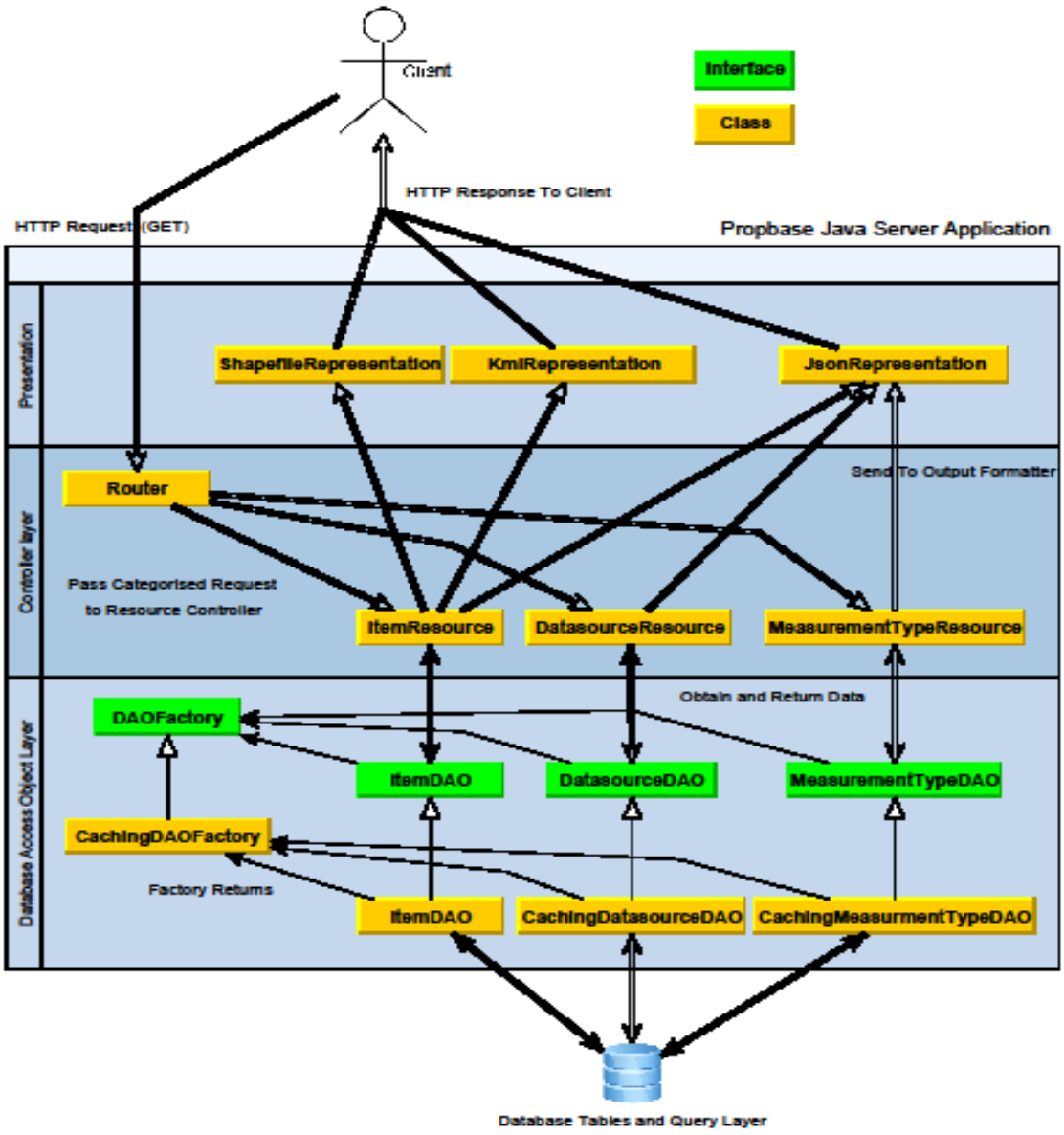
The application is largely discoverable from the Root URL which provides an HTML and JavaScript search implementation that can be converted into other languages, although this may be missing a couple of the more recent features. The html headers should also contain links to alternate representations. To access these, you are encouraged to try various requests to see what is returned. Content negotiation can be done either by using the HTTP accepts header or by specifying a file extension at the end of the last URL segment.

Architecture Diagram



Appendix

PropBase Server Internals



Available URL patters are composed of ([] indicates optional components - | indicates an either or option, / or : are literals that will need to be included in the URL)

<root> = 'http://bgsintranet/JavaApps/propbase/' – Root URL for service (You should be able to discover stuff from here)

<q> String variable – Free text filter.

<start> Integer variable number for first record to return (default = 0)

Martin Nayembil, Anne Richardson, Graham Smith, Simon Burden, 06/03/2014

<size> Integer variable number of records to return (default = 10)

<item> = items

<items> = <item>[?q=<q>][(&|?)start=<start>[&size=<size>]] – Filter to return propbase measurement item(s).

<dataSource> = datasources

<dataSources> = <dataSource>[?start=<start>[&size=<size>]] – Filter to return propbase dataSource(s)

<type> = types

<types> = <types>[?start=<start>[&size=<size>]] – Filter to return propbase measurement type(s)

<itemId> = integer variable - Id of a propbase measurement item.

<dataSourceId> = String variable - Id of a datasource.

<typeId> = String variable - Id of a propbase measurement type.

<dataSourceSet> = <dataSource> / <dataSourceId> [:<dataSourceId>[:<dataSourceId>]] (not limited to 3) – Filter by multiple datasources.

<typeSet> = <type> / <typeId> [:<typeId>[:<typeId>]] (not limited to 3) – Filter by multiple measurement types.

<x> <y> <z> = double variables – coordinates in three dimensions.

<crs> = integer variable – coordinate reference system the associated coordinates are in, currently limited to 27700 (BNG – default) and 4236 (WGS-84)

<location> = [<crs>::]<x>,<y>[,<z>]:<x>,<y>[,<z>] – Filter spatially using a bounding box or volume

Home Page and HTML Search Form = <root>

All features in category = <root> (<items> | <dataSources> | <types>)