iDR1: Example Queries for the GES archive

This document pertains to release iDR1 of the GES archive. It should not be used with subsequent releases of the archive; some aspects of the discussion and examples are incorrect for subsequent releases.

This note presents a number of examples of typical queries of the GES archive. The examples are intended to be similar to the queries which users new to the archive will often make.

Prerequisites

These examples assume familiarity with both the rudiments of SQL and use of the GES SQL query pages.

A beginner's introduction to SQL is available here.

Instructions for using the GES SQL query pages are available here.

The examples are divided into a number of sections based on the type of query being performed:

Queries on Target Stars

These queries return lists of target stars selected from the archive. Note that they do *not* return lists of spectra of those stars: a given star may be observed more than once. Also the physical parameters of stars (effective temperature *etc.*) are dependent on the spectrum from which they are derived (the star may vary), so such queries must also be made on the spectra; see the following sections for examples.

Aside on star names

The naming of stars, in catalogues and elsewhere, can be surprisingly problematic, with the brighter stars, in particular, enjoying a variety of designations. Some catalogue naming schemes derive a star's name from its celestial coordinates, others do not; there are pros and cons to both approaches.

In the GES survey each star is assigned a name based on its celestial coordinates. This name is formed by concatenating the sexagesimal Right Ascension in hours with the sexagesimal Declination in degrees according to the following scheme:

```
hhmmssss±ddmmsss
```

where ± specifies the sign of the Declination, either '+' or '-'. Several of the tables in the archive include the name of the target stars in this format, usually in a column called 'cName'. The names tabulated in column 'cName' are derived from each star's equatorial coordinates as supplied by ESO. Currently the following tables contain a 'cName' column:

```
Target
Spectrum
AstroAnalysis
```

1. **Is a given star in the archive?**

A list of all the stars in the GES archive is stored in the 'Target' table. This table contains two columns of star names. One is column 'cName', as described above, which contains the name derived from the star's equatorial coordinates. The other column is called 'object' and contains the star's name as supplied by ESO, in the same format as 'cName'. Usually the two names will be the same for a given star, but they may sometimes differ in the least significant digit in either coordinate. The following queries select all the information in table 'Target' for star 11053303-7700120 using the 'cName' and 'object' column respectively (for this star the two names are the same):

```
SELECT * FROM Target WHERE cName ='11053303-7700120';

SELECT * FROM Target WHERE object='11053303-7700120';
```

2. **Which (if any) of the following list of stars are in the archive?**

   If you wish to check which of the stars in a list are in the archive then the SQL 'IN' clause can be used. The following queries show the use of this clause for selections on both the 'cName' and 'object' columns. Note that in principle there is no restriction on the size of the list. Also note that only stars in the list that are found in the archive will be listed; any which are not in the archive will not appear (star 'bad-name' is included in the example to illustrate this point).

   ```
   SELECT * FROM Target WHERE cName IN('11034945-7700101', '11044460-7706240',
   'bad-name', '11053303-7700120');

   SELECT * FROM Target WHERE object IN('11034945-7700101', '11044460-7706240',
   'bad-name', '11053303-7700120');
   ```

   Any stars not in your list can be identified using a query of the (slightly cumbersome) form:

   ```
   SELECT * FROM ( VALUES ('11034945-7700101'), ('11044460-7706240'), ('bad-
   name'), ('11053303-7700120')) AS starList(starName) WHERE starName NOT IN
   (SELECT cName FROM Target);
   ```

3. **Select all the stars within a given range of RA and Dec.**

   The J2000 Right Ascension and Declination of stars in the GES survey are respectively stored in columns 'ra' and 'dec' of table 'Target'. Both coordinates are stored in decimal degrees (note in particular that the Right Ascension is *not* stored in hours) and southern declinations are negative. Thus, the the first step is to convert the coordinates of your range to this format. The stars enclosed in the region can then be found with a query of the form:

   ```
   SELECT * FROM Target WHERE (ra BETWEEN ra_min AND ra_max) AND (dec BETWEEN
   dec_min AND dec_max);
   ```

   Suppose your range was 70 to 80 degrees of Right Ascension and -45 to -30 degrees of Declination then the corresponding SQL would be:

   ```
   SELECT * FROM Target WHERE (ra BETWEEN 70 AND 80) AND (dec BETWEEN -45 AND
   -30);
   ```

Queries on Spectra

These queries return lists of spectra extracted from the GES archive. Recall that each star may have more than one spectrum (because it was observed more than once).

1. **Which (if any) spectra are available for a given star?**

   All the spectra in the GES archive are listed in table 'Spectrum'. This table contains auxiliary information about each spectrum (metadata in the jargon of computer science), such as a magnitude estimate for the target star, the observing priority, the exposure time, *etc*. For convenience of identifiying stars a 'cName' column is included listing each star's name. Other information pertaining to the target star, rather than to the spectrum, can be listed by joining the 'Spectrum' and 'Target' tables. The spectra themselves are not stored in a database table but as FITS files. As a first example, to list all the spectra for star 11053303-7700120:

   ```
   SELECT * FROM Spectrum WHERE cName ='11053303-7700120';
   ```

2. **Which (if any) spectra are available for a given list of stars?**

   Queries to list all the spectra available for a list of stars are similar to those for a single star (above) but use the 'IN' clause to specify the star list. Note that in principle there is no restriction on the size of the star list. Also note that only stars in the list that are found in the archive will be listed; any which are not in the archive will not appear (star 'bad-name' is included in the example to illustrate this point). The following query lists all the spectra available for a list of stars:

   ```
   SELECT * FROM Spectrum WHERE cName IN('11034945-7700101', '11044460-7706240',
   'bad-name', '11053303-7700120');
   ```

   Any stars with no spectra can be identified using a query of the (slightly cumbersome) form:

   ```
   SELECT * FROM ( VALUES ('11034945-7700101'), ('11044460-7706240'), ('bad-
   name'), ('11053303-7700120')) AS starList(starName) WHERE starName NOT IN
   (SELECT cName FROM Spectrum);
   ```

3. **How many spectra are available for each of a given list of stars?**

   The following query counts the number of spectra for each of the stars in a list. Note that any stars for which there are no spectra are not included in the returned list (star 'bad-name' is included in the example to illustrate this point).

   ```
   SELECT cName AS 'star', COUNT(cName) AS 'no. of spectra' FROM Spectrum WHERE
   cName IN ('11034945-7700101', '11044460-7706240', 'bad-name',
   '11053303-7700120') GROUP BY cName ORDER BY cName;
   ```

   See the previous example for how to check for any stars in the list which have no spectra.

4. **List all the spectra available for all the stars within a given range of RA and Dec.**

   As previously described queries to find all the stars in a given region of Right Ascension and Declination must be made on the 'Target' table because it is the only table to list each star's celestial coordinates. The 'Target' table can then be joined with the 'Spectrum' table to find the spectra available for each star. For example, to list the name, equatorial coordinates, exposure time, signal-to-noise ratio and magnitude of all the spectra for all the stars in the range 70 to 80 degrees of Right Ascension and -45 to -30 degrees of Declination:

   ```
   SELECT
     tg.cname, tg.ra, tg.dec, sp.expTime, sp.snr, sp.mag
   FROM
     Target tg,
     Spectrum sp
   WHERE tg.targetId = sp.targetId
     AND (tg.ra BETWEEN 70 AND 80)
     AND (tg.dec BETWEEN -45 AND -30);
   ```

Internal identifiers in GES archive tables

The above query joined the 'Target' and 'Spectrum' tables. This operation could have been done using the 'cName' column to identify corresponding rows in the tables since this column occurs in both tables. However, it was actually done using a column called 'targetId' which also occurs in both tables. These 'targetId' columns have a similar purpose to 'cName': to uniquely identify a given star (or 'target') and hence to allow joins to proceed by identifying rows in the two tables pertaining to the same star.

The advantages of using column 'targetId' rather than column 'cName' are that the former is numeric and indexed, which permits fast and efficient access. Consequently 'targetId' should be used in preference to 'cName' to identify corresponding stars in two or more tables. However, you should never need to know the actual values of 'targetId' (they are just numbers) as they are invented internally for the archive and have no wider astronomical significance. Moreover, they are created afresh for each release of the database, so *do not* rely on them remaining the same in different releases. For example, if you make a note of the 'targetId' of your favourite star in the current release and then query the following release, when it appears, using this value you will retrieve data for a different star.

The archive contains several other identifiers in addition to 'targetId'. For example, every row in the

'Spectrum' table also has an identifier, column 'specId', which uniquely identifies it. This identifier is used, for example, in identifying the spectrum from which individual astrophysical analysis parameters (effective temperature *etc.*) have been derived. Several different types of identifier will appear in the examples below.

5. **Which (if any) spectra are available for a given list of stars and with which spectrograph were they obtained?**

The 'Spectrum' table does not include a column listing the spectrograph with which it was acquired, so a more involved approach is required to show the instrument used.

GES spectra are all obtained using one of two multi-object spectrographs: Giraffe and UVES. Such instruments acquire a group (or 'frame') of spectra simultaneously. The 'SpecFrame' table lists all such frames included in the GES survey and contains details pertaining to the entire frame rather than individual spectra. It includes column 'instrument' which lists the spectrograph used. The values corresponding to the the two instruments are: 'GIRAFFE' and 'UVES' (note that the values are in upper case). To show the spectrograph used the rows selected from the 'Spectrum' table must be joined to the rows for their parent frames in the 'SpecFrame' table. The 'SpecFrame' table has a column, 'specFrameId', which contains an integer value that uniquely identifies each specFrame (see the note about identifiers, above). Table 'Spectrum' also contains a 'specFrameId' column and it lists the identifier of the specFrame from which each spectrum was extracted. Thus:

```
SELECT
  spec.cName, frame.instrument
FROM
  Spectrum spec,
  SpecFrame frame
WHERE spec.specFrameId = frame.specFrameId
  AND spec.cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414');
```

A similar query can be used to show only the spectra obtained with a given spectrograph:

```
SELECT
  spec.cName, frame.instrument
FROM
  Spectrum spec,
  SpecFrame frame
WHERE spec.specFrameId = frame.specFrameId
  AND spec.cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414')
  AND frame.instrument = 'UVES';
```

It is also possible to combine the 'Spectrum', 'SpecFrame' and 'Target' tables to list all the spectra available in a given range of Right Ascension and Declination and to show the spectrograph with which they were observed:

```
SELECT
  tg.cname, tg.ra, tg.dec, frame.instrument, sp.expTime
FROM
  Target tg,
  Spectrum sp,
  SpecFrame frame
WHERE tg.targetId = sp.targetId
  AND sp.specFrameId = frame.specFrameId
  AND (tg.ra BETWEEN 70 AND 80)
  AND (tg.dec BETWEEN -45 AND -30);
```

The values of column 'instrument' in the 'SpecFrame' table corresponding to the two spectrographs used are 'GIRAFFE' and 'UVES', however to check which values actually occur in the table:

```
SELECT DISTINCT instrument FROM SpecFrame;
```

or to check the number of times each occurs and to order the resulting list alphabetically by instrument:

```
SELECT instrument, COUNT(instrument)
FROM   SpecFrame
GROUP BY instrument
ORDER BY instrument;
```

Queries on Recommended Analyses

Aside on recommended and individual parameters

This discussion applies to release iDR1; subsequent releases may be different and previous releases were different.

The spectra acquired as part of the GES survey are reduced and analysed by the GES Consortium. The structure of the Consortium includes several **working groups** and each working group typically comprises several **nodes**.

Working groups

typically perform a single type of analysis, usually for stars of a given spectral class (or range of spectral classes) and/or for a given spectrograph (Giraffe or UVES). The actual analyses are performed by the nodes comprising the woking group. The iDR1 release contains results from working groups WG10 to WG13.

Nodes

within a working group perform an individual analysis on the stars assigned to the working group or a subset of them. The nodes use a variety of analysis methods to arrive at their sets of parameters and abundances.

Every analysis produces a standard set of astrophysical parameters. Some of these parameters pertain to the entire spectrum (such as the effective temperature or surface gravity) while others pertain to the analysis of individual species (typically abundances). Thus, for every spectrum that it works on each working group will assemble a set of astrophysical parameters with one set from each of its nodes (or some subset of them). The working group will combine these results to produce a 'best' or 'recommended' set of astrophysical parameters for each spectrum. The archive contains both these recommended parameters and the individual parameters computed by the various nodes. This section gives examples of querying the recommended parameters and the following section gives examples of querying individual parameters. *However, you should be aware of, and must conform to, Consortium policy as it pertains to the use of parameters and abundances in publications and calculations that lead to publications.*

1. **List the recommended analysis parameters for all the spectra of a given star**

   All the recommended parameters determined for stars in the archive are available in view 'RecommendedAstroAnalysis' (a view is an entity in a relational database which for the present purposes behaves like a table). The columns in view 'RecommendedAstroAnalysis' include 'cName' which lists the name of the star observed and 'wg' which lists the name of the working group that performed the analysis (though these names are not very informative, being of the form 'WGxx', for example 'WG10'). To list selected parameters (here effective temperature, log surface gravity, FeH ratio, li1 abundance and C1 abundance) for all the spectra observed for a given star and the name of the working group that performed the analyses:

   ```
   SELECT cName, wg, teff, logg, feh, li1, c1
   FROM   RecommendedAstroAnalysis
   WHERE  cName ='11053303-7700120';
   ```

   Note that two records are returned as this star has been analysed by WG11 and WG12; in general a given spectrum may be analysed by more than one working group.

2. **List the recommended analysis parameters for all the spectra available for a list of stars**

Similarly to list selected parameters for all the spectra observed for a list of stars:

```
SELECT cName, wg, teff, logg, feh, li1, c1
FROM   RecommendedAstroAnalysis
WHERE  cName IN ('11034945-7700101', '11044460-7706240', '21101955-0200414')
ORDER BY cName;
```

The list is ordered by 'cName' so that results for the same star appear together.

3. **List recommended analysis parameters alongside information about the spectrum from which they were obtained.**

   To list selected parameters determined from a spectrum alongside information about that spectrum it is necessary to join the 'RecommendedAstroAnalysis' and 'Spectrum' tables:

```
SELECT
  racc.cName, racc.wg, racc.teff, racc.logg, racc.feh,
  sp.expTime, sp.snr
FROM
  RecommendedAstroAnalysis racc,
  SpectrumGroup spg,
  Spectrum sp
WHERE racc.specGroupId = spg.specGroupId
  AND spg.specId = sp.specId
  AND racc.cName ='11053303-7700120';
```

   In this example in addition to the working group and some recommended parameters (effective temperature, log surface gravity and FeH ratio) the exposure time and signal-to-noise ratio of the spectrum from which they were derived are listed.

Aside on linking recommended parameters to their parent spectra

You will have noticed that in the above example rather than joining the 'RecommendedAstroAnalysis' and 'Spectrum' tables directly they were joined indirectly through table 'SpectrumGroup', which has not been mentioned previously. The reason for needing this table is as follows. Usually recommended parameters are derived from a single spectrum. However, in some cases they are derived from two spectra, typically covering different wavelength regions. To cope with these latter cases the notion of the 'spectrum group' was introduced.

A spectrum group is simply a unique group of spectra from which recommended parameters have been derived. Most spectrum groups will consist of a single spectrum, but some will consist of two (as, indeed, the above example does). Each spectrum group has a unique identifier, which is tabulated in 'RecommendedAstroAnalysis' as column 'specGroupId'. Table 'SpectrumGroup' lists the spectrum identifiers of the spectra that constitute each spectrum group. If the spectrum group contains two spectra then the query will return two rows for each recommended parameter, one for each spectrum. It is a little easier to see what is going on if the 'specGroupId' and 'specId' identifiers are included in the listing (though, of course, they have no significance outside the GES archive):

```
SELECT
  spg.specGroupId, sp.specId,
  racc.cName, racc.wg, racc.teff, racc.logg, racc.feh,
  sp.expTime, sp.snr
FROM
  RecommendedAstroAnalysis racc,
  SpectrumGroup spg,
  Spectrum sp
WHERE racc.specGroupId = spg.specGroupId
  AND spg.specId = sp.specId
  AND racc.cName ='11053303-7700120';
```

In any event, the upshot is that the 'RecommendedAstroAnalysis' and 'Spectrum' tables must be linked using the 'SpectrumGroup' table.

It is, of course, possible to display additional information about the spectra by joining the 'RecommendedAstroAnalysis', 'SpectrumGroup' and 'Spectrum' tables with the 'SpecFrame' table in the same way that the 'Spectrum' and 'SpecFrame' were joined (above). For example, to include the spectrograph used to acquire the spectra in the above query:

```
SELECT
  racc.cName, racc.wg, racc.teff, racc.logg, racc.feh,
  sp.expTime, sp.snr,
  frame.instrument
FROM
  RecommendedAstroAnalysis racc,
  SpectrumGroup spg,
  Spectrum sp,
  SpecFrame frame
WHERE racc.specGroupId = spg.specGroupId
  AND spg.specId = sp.specId
  AND sp.specFrameId = frame.specFrameId
  AND racc.cName ='11053303-7700120';
```

Queries on Individual Analyses

As described in the previous section each working group comprises several nodes and each node (or a sub-set of them) performs an independent analysis of each spectrum (or rather each spectrum group). The working group then combines these individual results to derive a set of recommended parameters for the spectrum group and these are tabulated in view 'RecommendedAstroAnalysis'.

In addition to the recommended parameters the parameters derived from the analysis of the individual nodes are also available in the GES archive. These individual parameters are listed in table 'AstroAnalysis' whose columns are identical to those of view 'RecommendedAstroAnalysis'. Where a query of 'RecommendedAstroAnalysis' would yield a single row listing recommended parameters the corresponding query of 'AstroAnalysis' would yield a set of rows, each row corresponding to an analysis by an individual node. The node responsible for the analysis is listed as column 'nodeName'. Note that the recommended parameters are also included in table 'AstroAnalysis' and have a 'nodeName' of the name of the working group. For example, to list selected parameters (here effective temperature, log surface gravity, FeH ratio, li1 abundance and C1 abundance) for all the spectra observed for a given star and the name of the working group and node that performed the analyses:

```
SELECT cName, wg, nodeName, teff, logg, feh, li1, c1
FROM   AstroAnalysis
WHERE  cName ='11053303-7700120'
ORDER BY wg, nodeName;
```

Note that the results are sorted by column 'wg' and then 'nodeName' so that the results for each working group appear together and within each working group the nodes are listed alphabetically. In a similar fashion, all the queries in the previous section can be changed to show the individual rather than recommended parameters by replacting view 'RecommendedAstroAnalysis' with table 'AstroAnalysis' and adding the 'ORDER BY' clause.

Queries on the Atomic and Molecular Line Lists

The GES archive contains a set of atomic and molecular line lists. Atomic lines both with and without hyperfine splitting are included. These data are used consistently in all analyses of the GES spectra carried out within the GES Consortium. All the lines available are listed in table 'LineList', which contains in excess of 33 million rows, most of them corresponding to molecular lines. Several views containing useful sub-sets of the line list data are also provided. In many cases these views are likely to be more convenient to use than the complete 'LineList' table. The views available are as follows: (table 'LineList' is included for convenience and completeness):

View	Description	Rows
LineAtomHfs	Atomic lines with hyperfine splitting (HFS) only.	61,124
LineAtomNoHfs	Atomic lines without hyperfine splitting (HFS) only.	59,836

LineMol	Molecular lines only.	33,154,740
LineMolAtomHfs	Molecular lines and atomic lines with hyperfine splitting (HFS).	33,215,864
(LineList)	The complete set of atomic lines with and without hyperfine splitting and molecular lines.	33,275,700

The following examples all present queries of view 'LineAtomHfs' (that is atomic lines with hyperfine splitting). However, all the views, and table 'LineList' have the same set of columns so the examples can be applied to any of them by simply substituting the appropriate view or table name.

1. **Which lines are available in a given wavelength range?**

   Column 'lambda' lists the wavelength in Ångström. Thus, to see all the lines in wavelength range 4800 to 4810 Å:

   ```
   SELECT * FROM LineAtomHfs
   WHERE lambda BETWEEN 4500 AND 4800
   ORDER BY lambda ASC;
   ```

   Note the inclusion of the 'ORDER BY' clause so that the lines are listed in order of increasing wavelength.

2. **Which lines are available for a given species?**

   Column 'name1' lists the element giving rise to each line and column 'ion' the ionisation state of each species. Thus, to see all the lines for aluminium 1, ordered by increasing wavelength:

   ```
   SELECT * FROM LineAtomHfs
   WHERE name1 = 'Al' and ion = 1
   ORDER BY lambda ASC;
   ```

3. **Which species are available for a given element?**

   To see a list of the iron species available and the number of lines included for each species:

   ```
   SELECT name1, ion, count(ion) FROM LineAtomHfs
   WHERE name1 = 'Fe'
   GROUP BY name1, ion
   ORDER BY name1, ion;
   ```

4. **Which lines are available for a given element?**

   To see all the aluminium lines, for example, ordered by increasing wavelength:

   ```
   SELECT * FROM LineAtomHfs
   WHERE name1 = 'Al'
   ORDER BY lambda ASC;
   ```