OPUS_SQL_GEN

Shaw-Hong Kao
November 27, 1996

Usage

To generate load opus sql output file, at command line, type :

```
$ opus_sql_gen [p1]
  where p1 = delivery number, if ommit, use default delivery_number.value
```

It will create output file opus_<N>_<I>.sql, where <N> is delivery number, <I> is instrument abbreviation.

Purpose

The tool retrieve CDBS database's OPUS data and generate a script file opus_<N>_<I>.sql containing insert/delete sql for use by OPUS.

Description/Algorithm

1. The tool first check input arguments, if user did not provide <p1> (delivery_number), the tool will use delivery_number.value as default delivery_number.
2. A script(output) file will be create, start with "begin transcation".
3. Create local variable @load_date to save current time "dd-MMM-yyyy hh:mm:ss" for use by load_time fields in OPUS.
4. Write information message "LOAD_DATE_USED=<current datetime>" and "DELIVERY_NUMBER_USED=<delivery_number>" to output.
5. Loop through `opus_log` table to get `file_name`, `expansion_number`, `instrument`, `reference_file_type`, `operation` data having `delivery_number` = delivery_number
6. Get `opus_table` name from `relation_info` table having `instrument`, `reference_file_type` equals to the data retrieved from item 5.
7. Create map record array which contains cdbs table, fields name and corresponding opus table, fields name by :

   ```
   select <cdbs_field>, <cdbs_table>, <opus_field>
     from opus_fields
    where opus_table = <opus_table> and cdbs_table like "<instrument>%"
   ```

   to get field data type by :

```
select column_name = c.name, type = t.name
  from dbo.syscolumns c, dbo.systypes t, dbo.sysobjects o
 where o.name = <cdbs_table> and c.name = <cdbs_field>
   and c.id = o.id and o.type = 'U' and c.usertype *= t.usertype
```

8. Based on the map array, create a string <sl> for the select-list:

   ```
   <map[0].cdbs_table_name>.<map[0].cdbs_field_name> s0,
   <map[1].cdbs_table_name>.<map[1].cdbs_field_name> s1,
   <map[2].cdbs_table_name>.<map[2].cdbs_field_name> s2, ...
   ```

   and run query to get cdbs data:

   ```
   select distinct <sl>
     from <instrument>_file, <instrument>_row
    where <instrument>_file.file_name = <instrument>_row.file_name
      and <instrument>_file.file_name = <file_name>
      and <instrument>_file.expansion_number = <instrument>_row.expansion_number
      and <instrument>_file.expansion_number = <expansion_number>
   ```

9. If `operation` data (from item 5) = "I" then generate insert statement and write to output:

   ```
   insert <opus_table> (map[i].opus_field..., load_time[, instrument])
   values (map[i].field_value,..., @load_date[, "<instrument_abbrev>")
   ```

   note: <instrument> data is for OPUS <caltable> table only

10. If operation data = "R" or "D" then generate a delete statement and write to output:

    ```
    delete from <opus_table>
     where map[i].opus_field = map[i].field_value ...
      [and instrument = "<instrument_abbrev>"]
    ```

11. Go back to item 5) until no more record returned from `opus_log` table.

12. Write information message "OPUS_MODS_OK" and "commit transaction" to output and close database connection and output file.

Files

There are two files for this tool, a source file write in C, using STDB interface to retrieve database data, a make file to generate executable:

```
opus_sql_gen.c -- source code
Makefile
```

Testing

```
$ # use default CDBS db account and default database to load test data to
$ # CDBS database and run opus_sql_gen to create load opus sql files
$ test_opus_sql_gen
```

Load SQL File

```
$ # use default OPUS db account and default OPUS database
$ isql -S<server> -i<sql file>
example:
$ isql -SDANEEL -iopus_104_W.sql
```