Regional Cooperation for
Limited Area Modeling in Central Europe

LACE
nwp central europe

ODB & MANDALAY

Alena Trojáková

DHMZ SHMU ARSO METEO Slovenia OMSZ ZAMG ROMANIA ANM

ODB

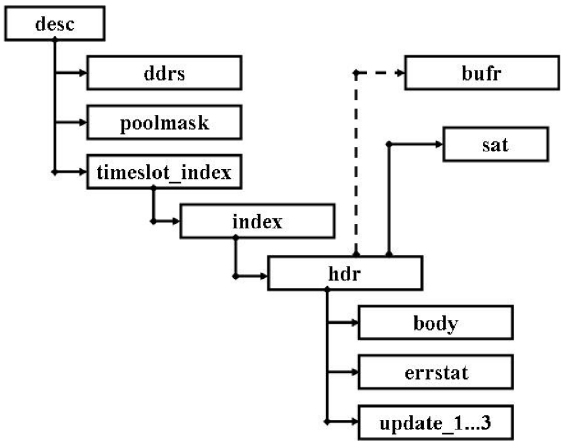
- **Observational DataBase (ODB) is hierarchical database software developed at ECMWF to manage very large observational data volumes**
- **ODB components:**
  - **ODB/DDL Data Definition Language (flexible data layout definition of database)**
  - **ODB/SQL query language (fast data retrieval)**
  - **ODB Fortran90 interface layer (data manipulation as create, update and remove, execution of sql-queries and retrieval of data, control of MPI and/or OpenMP-parallelization)**
- **ODB content:**
  - **observation identification information (date, position, station ID)**
  - **observed values**
  - **various flags indicating quality and validity of an observation (active,blacklisted,...)**
  - **departure from observed value (obs-guess, obs-analysis)**
  - **bias corrections, satellite specific information like zenith angle, field of view, ...**
  - **other important observational processing and meteorological information**

ODB

- **ODB structure**

  - basic building blocks called table (can be seen as a matrice (2D-array)) with a number of rows and columns containing numerical data (example hdr: general information of one report (date, time, station ID))

- **data are organized into a *tree-like* structure**



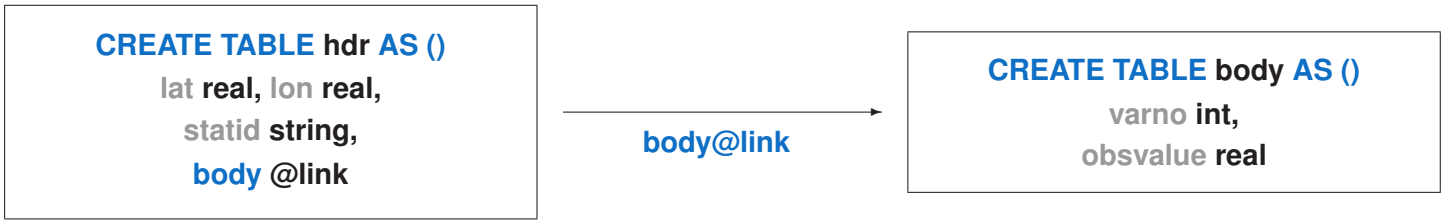
- **structure allows ”repeating” information using parent/child relationship: each parent can have many children but each child only has one parent**

ODB/DDL - Data Definition Layout

LACE
nwp central europe

- DDL file defines the structure (hierarchy)
- ASCII file
- consists of uniquely named TABLEs
- tables are made up of uniquely named COLUMNs
  notation: `column_name@table_name`
- each COLUMN has a specific type
  - integer/real/string
  - packed data type
  - YYYYMMDD, HHMMSS (storage of date)
  - bitfield type (maximum 32 one-bit members per type,
    notation: `column_name.bitfield_name@table_name`
  - @LINK to define connections between TABLEs

```
CREATE TABLE table_name AS ()
  column_name1 data_type1,
  column_name2 data_type2,
```



DHMZ ARSO METEO Slovenia ZAMG ROMANIA ANM

ODB/SQL - data retrieval

LACE nwp central europe

- **data extraction by query language ODB/SQL via so-called views**

  [CREATE VIEW view_name AS]
  SELECT [DISTINCT] column_name (s)
  FROM table(s)
  WHERE cond ORDERBY sort_column_name(s) [ASC/DESC]

can be used in an interactive way via ODB-tools (odbsql,...)

Examples:

- **find distinct values of obstype and sort them DESCending**

```
select distinct obstype from hdr orderby obstype desc
```

- **vertical profile of MEAN and STD for O-G for sensor HIRS**

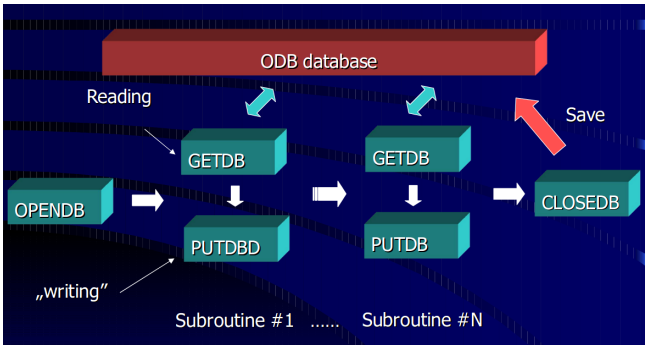
```
select count(*), satid,obstype,varno,sensor,press,avg(fg_depar),stdev(fg_depar)
from hdr,body,sat
where obsvalue is not NULL and status.active@body = 1 and sensor = 0
```

- **find location and values of all active SYNOP observations**

```
select lat,lon,obsvalue from hdr,body where obstype = 1 and status.active@body
```

ODB Fortran90 interface

- **layer to provide database access to:**
  - **- open & close database**
  - **- attach to & execute precompiled ODB/SQL queries**
  - **- load, update & store queried data**
  - **- inquire information about metadata**



Credit: S.Kertesz ALADIN maintenance wksp 2002

- **allow use MPI**
- **TABLEs are devided into so called "pools" between processors, (usually number of pools equals to number of MPI tasks)**
- **each query need to be pre-compiled/linked with the main user program**

view (.sql file)	→	ODB/SQL compiler	→	C-source (.c file)	→	C compiler	→	object (.o file)

- **each ARPEGE/IFS cycle has its own ODB version !**

Practical aspects

Practical aspects

LACE
nwp central europe

- ODB usage in ARPEGE/ALADIN:
  - ALDODB - master for configuration 002,131,701
  - BATOR - master for ODB creation
  - ODBTOOLS - master for ODB manipulation
  - MANDALAY - master for ODB conversion to ASCII
- each query need to be pre-compiled/linked with the main user program
- each ARPEGE/IFS cycle has its own ODB version !
- ODB content:
  - observation identification information (date, position, station ID)
  - observed values
  - various flags indicating quality and validity of an observation (active,
  - departure from observed value (obs-guess, obs-analysis)
  - bias corrections, satellite specific information like zenith angle, field of view, ...
  - other important observational processing and meteorological information

ODB in ARPEGE/ALADIN source code

LACE
nwp central europe

- **OPENDB - opens ECMA/CCMA databases**
- **GETDB**
  - execute one or more SQL queries (as defined in ctxinitdb.F90
  - calls ODB_select, allocates matrices ROBHDR,ROBODY,...
  - then calls ODB_get to fill out the observational matrices
    - ROBHDR: index & hdr - tables related data
    - ROBODY: body, errstat, update,.. - tables related data
    - MLNKH2B: coupling between ROBHDR & ROBODY

```
HDR_LOOP: do jobs=1, NROWS_ROBHDR
  ROBHDR(jobs,MDBLAT) = <some_thing>
  BODY_LOOP: do jbody= MLNKH2B(jobs), MLNKH2B(jobs+1) - 1
      if ( ROBODY(jbody,MDBVNM) == <varno> ) then
          ROBODY(jbody, MDBOMF) = <some_thing>
      endif
    enddo BODY_LOOP
enddo HDR_LOOP
```

- **PUTDB**
  -returns the contents of the updated matrices bask to (in-memory) database data structures via routine ctxputdb.F90
  - calls ODB_put, deallocates matrices and calls ODB_cancel
- **CLOSEDB - closes ECMA/CCMA databases**

ODB in ARPEGE/ALADIN source code

LACE
nwp central europe

- **correspondence of ARPEGE/IFS variables and ODB/SQL:**

```
INTEGER(KIND=JPIM) ::  mdbdat   !  'date@hdr'
INTEGER(KIND=JPIM) ::  mdbrfl   !  'report_rdbflag@hdr'
INTEGER(KIND=JPIM) ::  mdbrst   !  'report_status@hdr'
INTEGER(KIND=JPIM) ::  mdbrev1  !  'report_event1@hdr'
INTEGER(KIND=JPIM) ::  mdbrble  !  'report_blacklist@hdr'
INTEGER(KIND=JPIM) ::  mdbsid   !  'statid@hdr'
INTEGER(KIND=JPIM) ::  mdblat   !  'lat@hdr'
INTEGER(KIND=JPIM) ::  mdblon   !  'lon@hdr'
INTEGER(KIND=JPIM) ::  mdbalt   !  'stalt@hdr'
...
INTEGER(KIND=JPIM) ::  mdbvnm   !  'varno@body'
INTEGER(KIND=JPIM) ::  mdbvar   !  'obsvalue@body'
INTEGER(KIND=JPIM) ::  mdbomn   !  'an_depar@body'
INTEGER(KIND=JPIM) ::  mdbomf   !  'fg_depar@body'
INTEGER(KIND=JPIM) ::  mdbflg   !  'datum_anflag@body'
...
```

- **for complete definitions see arpifs/common/yomdb_vars.h**
- **complete SQL queries see odb/ddl/*sql**

ODB applications - BATOR

- **BATOR - master for ODB creation**
- **ODB data are stored in directory structure**

```
ECMA.synop:
ECMA.synop/ECMA.dd ECMA.sch ECMA.flags IOASSIGN
ECMA.synop/1/body  conv_body  errstat  index poolmask update_1,2,3
            conv  desc  hdr modsurf  timeslot_index
ECMA.synop/2/body conv_body  errstat  index poolmask update_1,2,3
             conv  desc hdr modsurf timeslot_index
  ...
ECMA.synop/Npool/...
```

ODB applications - ODBTOOLS

LACE
nwp central europe

- **ODBTOOLS - master to perform various ODB manipulation (”shuffles”)**
  - data repartition
  - change of the number of the pool
  - timeslot and time-window definition
  - data selection

- **execution is controlled by a set of environmental variables:**
  `export ODB_IO_METHOD=1`
  `export ODB_CMA=`*database type definition*
  `export IOASSIGN=` *path to IOASSIGN file - the directory structure of the database*
  `export ODB_SRCPATH_ECMA` = *the location of ODB sub-bases' description files*
  `export ODB_DATAPATH_ECMA` = *the location of ODB sub-bases' data files*
- **stand-alone program, but more&more inlined within MASTERODB**
  `export ODB_MERGEODB_DIRECT=1`
- **Examples:**

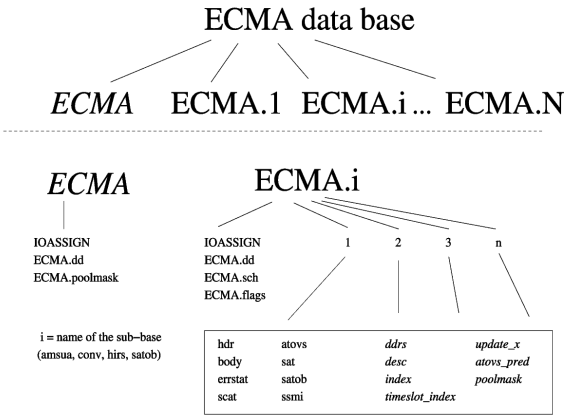
  ECMA $->$ CCMA translation (load balanced, active data, 131 database)

  CCMA $->$ ECMA update (”matchup”)

ODB applications - ODBTOOLS

LACE nwp central europe

- **ODB enables the preparation of separate ECMA "sub-bases" that can be handled as on common "virtual" ECMA database**
  - **more flexible for the users**
  - **each sub-bases has the same structure as ECMA database, but does not contain all the tables**
  - **"virtual" database has only descriptors pointing on the different sub-bases**



- **"merge" comprisis**
  - **creation of IOASSIGN file via script merge_ioassign**

    ```
    ./merge_ioassign -d $workdir -t sub-base1 -t sub-bases2 -t sub-bases3 ...
    ```

  - **a shuffle run - (creation of description files and adding missing TABLEs:** *update_x, atovs_pred, timeslot_index, index, desc, poolmask, ...*

    ```
    mpirun -np 1 ./shuffle -iECMA -oECMA -atotal_n_pools -b1
    ```

ODB browsing

LACE
nwp central europe

- **odbsql - ”dynamic” retrieval based**
  - **compilation is done on the fly**
  - **available in an ODB-standalone package only**

```
odbsql  -q 'select obstype,statid,lat,lon,varno from hdr,body '
```

- **mandalay - ”static” retrieval based**
  - **retrieval are based on predefined and user defined views (mandalay.sql):**

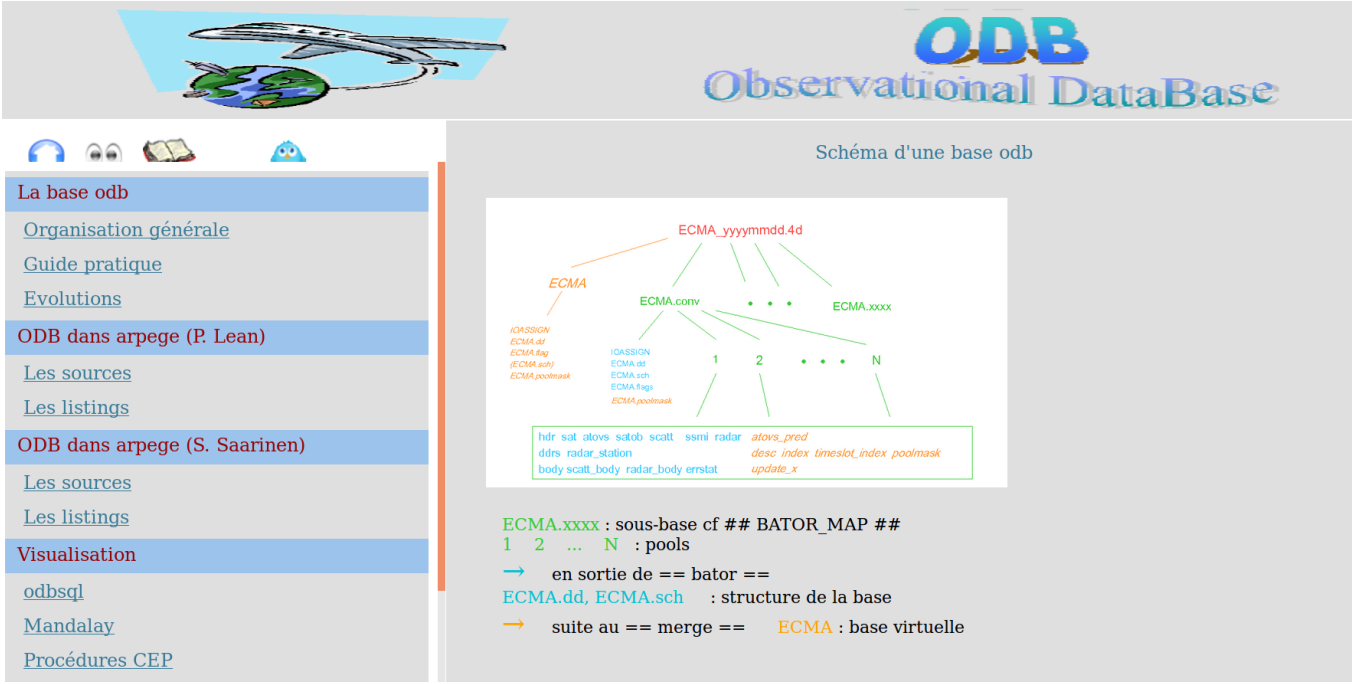
```
CREATE VIEW mandalay AS
SELECT
  obstype,statid,lat,lon,varno
FROM  hdr,body
```

  - **in case of change recompilation is needed (or a wrapper for re-compilation)**
  - **suitable for oper. application or frequently used request (observational monitoring,...)**
  - **export VERSION=1**
  - **export DEGRE=1**

```
mpirun -np 1 ./MANDALAY CMAFILE
```

ODB browsing hints

- **D. Puech: ODB documentation**
  http://www.umr-cnrm.fr/aladin/meshtml/DOC_odb/odb.php



ODB flags used by CANARI

- **Taillefer (2002): CANARI technical documentation**
  https://www.umr-cnrm.fr/gmapdoc/IMG/ps/canari_doc_cy25t1.ps
- **datum_anflag@body is coded over 29bits**

```
bits 1 to 4 : final quality code
bits 5 to 8 : first-guess quality code
bits 9 to 12 : spatial quality control code
bits 13 to 16 : variational quality code (not used in CANARI)
bits 17 to 20 : blacklist code
bit 21 : if set to 1, parameter used in the surface pressure analysis
bit 22 : if set to 1, parameter used in the wind and temperature analysis
bit 23 : if set to 1, parameter used in the relative humidity analysis
bit 24 : if set to 1, parameter used in the 2 meters temperature analysis
bit 25 : if set to 1, parameter used in the 2 meters relative humidity analysis
bit 26 : if set to 1, parameter used in the 10 meters wind analysis
bit 27 : if set to 1, parameter used in the precipitations analysis (not coded yet
bit 28 : if set to 1, parameter used in the snow analysis
bit 29 : if set to 1, parameter used in the SST analysis
```

```
odbsql  -q 'select statid,datum_anflag.ut2@body from hdr,body where varno == 39 '
```

ODB flags used by CANARI

- **datum_rdbflag@body coded over 30 bits, the first half concerns the quality of the vertical coordinate and the second half the quality of the parameter itself.**

```
bit 1 0 no human control
      1 human control
bit 2 0 no correction by the meteorological databank preprocessing
      1 correction by the meteorological databank preprocessing
...
bits 7 and 8 0 correct parameter versus previous analysis
             1 probably correct parameter versus previous analysis
             2 probably incorrect parameter versus previous analysis
             3 incorrect parameter versus previous analysis
bit 9 0 parameter no used by the previous analysis
      1 parameter used by the previous analysis
```

- **for complete definitions of bits see ./odb/ddl/type_definitions.h**

```
odbsql  -q 'select statid,"datum_rdbflag.*@body" from hdr,body where varno == 39 '
```

Acknowlegments

- **ECMWF training materials**
- **ALADIN maintenance & phasing training course**
  http://www.umr-cnrm.fr/gmapdoc/spip.php?article4
  http://www.umr-cnrm.fr/gmapdoc/spip.php?article208

Thank you for your attention !

Exercises

- find which observation types, variables and obs values are in your ECMA
- find blacklisted TEMP observations
- find observation errors for SYNOP meassurements at station 11518

Exercises

- **find which observation types, variables and obs values are in your ECMA**

```
select obstype,varno,obsvalue from hdr, body
```

- **find blacklisted TEMP observations**

```
select odbsql -q 'select statid from hdr,body where datum_status.blacklisted=1'
```

- **find observation errors for SYNOP meassurements at station 11518**

```
select odbsql -q 'select varno,obs_error from hdr,body,errstat
where statid = "11518"'
```