Preprocessing and format of observations for the data assimilation

Alena Trojáková
alena.trojakova@chmi.cz
Outline of the talk

• Observation
  – overview
  – preprocessing
  – format

• Overview of the ODB software

• ODB applications
  – BATOR
  – ODBTOOLS
  – ODB text browsing (MANDALAY, odbviewer)
Observation overview

- **conventional**
  - Surface & Marine - SYNOP, SHIP, TEMP, ..
    \((p, T_{2m}, R_{H2m}, v_{10m}, v_{10m}, RR, SST, ..)\)
  - Upper-air & Aircraft - TEMP, PROFILER, PILOT, AMDAR, SATOB, ..
    \((u, v, T, q, \phi)\)

- **satellite**
  - AIRS, AMSU-A/B, ASCAT, HIRS, IASI, MHS, SEVIRI, ..
    \((T_{b})\)

- **other platforms** (Doppler radar, solar radiation observations, ..)
Observation types in ARPEGE/ALADIN

<table>
<thead>
<tr>
<th>Variable</th>
<th>Value</th>
<th>Observation type</th>
</tr>
</thead>
<tbody>
<tr>
<td>NSYNOP</td>
<td>1</td>
<td>SYNOP, SYNOP.Ship, SYNOP.Ship</td>
</tr>
<tr>
<td>NAIREP</td>
<td>2</td>
<td>AIREP, AMDAR, ACAR, CODAR, COLBA</td>
</tr>
<tr>
<td>NSATOB</td>
<td>3</td>
<td>SATOB</td>
</tr>
<tr>
<td>NDRIBU</td>
<td>4</td>
<td>DRIBU, DRIFTER, BUOY, BATHY, TESAC, ERS1</td>
</tr>
<tr>
<td>NTEMP</td>
<td>5</td>
<td>TEMP, TEMP.Ship, TEMP.Drop, ROCOB</td>
</tr>
<tr>
<td>NPILOT</td>
<td>6</td>
<td>PILOT, PILOT.Ship, PILOTMOBIL, wind profiler</td>
</tr>
<tr>
<td>NSATEM</td>
<td>7</td>
<td>SATEM, TOVS</td>
</tr>
<tr>
<td>NPAOB</td>
<td>8</td>
<td>PAOB</td>
</tr>
<tr>
<td>NSCATT</td>
<td>9</td>
<td>scaterrometer(ERS)</td>
</tr>
<tr>
<td>NLIMB</td>
<td>10</td>
<td>GPS</td>
</tr>
<tr>
<td>NRADAR</td>
<td>13</td>
<td>radar</td>
</tr>
</tbody>
</table>

the observation types definition: obstype.h, yomcoctp.F90, sucmoctp.F90

<table>
<thead>
<tr>
<th>Value</th>
<th>variable name [unit]</th>
</tr>
</thead>
<tbody>
<tr>
<td>NVNUMB(1)</td>
<td>3 u-wind component [m/s]</td>
</tr>
<tr>
<td>NVNUMB(2)</td>
<td>4 v wind component [m/s]</td>
</tr>
<tr>
<td>NVNUMB(3)</td>
<td>1 geopotentiel [J/kg]</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
<tr>
<td>NVNUMB(96)</td>
<td>189 N2O</td>
</tr>
</tbody>
</table>

the variable definitions: varno.h, yomvnmb.F90, suvnmb.F90
the sensor definitions: sensor.h, yomtvrad.F90
Observation preprocessing

- preparation for use in NWP (data assimilation, verification, nowcasting, ...)
  - reception and storage
  - decoding and/or format conversion
  - local pre-treatment (generation of necessary parameters for advanced data (SEVIRI), initialization of various flags)
  - very basic quality control
  - conversion to the format suitable for NWP application (ODB)
program developed by Météo France (customization needed for other sites)

- read observation from **local database** and produce OBSOUL file
- handle SYNOP, AIREP, TEMP, GEOWIND, WindProfiler, ATOVS data
OBSOUL (ASCII)

- simple format
- suitable for testing of new observation types

```
date time
rec1
rec2
...
```

date: yyyymmdd
time: hhmmss
record: n header body1 ... bodyk

<table>
<thead>
<tr>
<th><strong>Header Description</strong></th>
<th><strong>Type</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>1 observation type (obstype@hdr)</td>
<td>integer</td>
</tr>
<tr>
<td>2 observation code</td>
<td>integer</td>
</tr>
<tr>
<td>3 latitude</td>
<td>real</td>
</tr>
<tr>
<td>4 longitude</td>
<td>real</td>
</tr>
<tr>
<td>5 station/satellite identification</td>
<td>character</td>
</tr>
<tr>
<td>6 date yyyymmdd</td>
<td>integer</td>
</tr>
<tr>
<td>7 time hh</td>
<td>integer</td>
</tr>
<tr>
<td>8 altitude</td>
<td>real</td>
</tr>
<tr>
<td>9 number of parameters (number of bodies)</td>
<td>integer</td>
</tr>
<tr>
<td>10 observation quality flags</td>
<td>integer</td>
</tr>
<tr>
<td>11 site dependant</td>
<td>integer</td>
</tr>
</tbody>
</table>
Observation formats in ARPEGE/ALADIN

<table>
<thead>
<tr>
<th>Body Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 type of parameter (varid@body)</td>
<td>integer</td>
</tr>
<tr>
<td>2 first vertical coordinate</td>
<td>real</td>
</tr>
<tr>
<td>3 second vertical coordinate (if necessary of other)</td>
<td>real</td>
</tr>
<tr>
<td>4 observed or measured parameter</td>
<td>real</td>
</tr>
<tr>
<td>5 parameter quality flag</td>
<td>integer</td>
</tr>
</tbody>
</table>

Examples:

SYNOP
42 1 10000014 50.01700 14.45000 '11520' 20100915 90000 304.0000 6 1111 100000
1 -101220.0 1.7000000E+38 0.0000000E+00 2064 39 97680.00 1.7000000E+38 288.8600 2048
58 97680.00 1.7000000E+38 71.00000 2048 7 97680.00 1.1426964E-03 8.0968356E-03 2048
41 97680.00 4.000000 260.0000 2048 91 97680.00 1.7000000E+38 80.00000 2048

AMDAR
22 2 10031144 67.60500 105.87334 LH715 20100915 83400 10600.00 2 11111 0
2 10600.00 1.7000000E+38 229.5000 4111 3 10600.00 6.200000 256.0000 4111
Observation formats in ARPEGE/ALADIN

**BUFR** (Binary Universal Form for data Representation) - has been designed to achieve efficient exchange and storage of data. It is self-defining, table-driven and very flexible.

- **BUFR** - (FM-94 BUFR)
- used for most of the satellite data
- key routine bator_decodbufr.mod.F90
- a configuration file to decode bufr-files (param_bator.cfg)

```
BEGIN amsub
  1 1 0 14
  codage  1  310010
  control  1  5  nb de canaux
  values  7  001007  Satellite identifier
  values 11  005041  Scan line number
  values 12  005043  Field of view number
  values 22  005001  Latitude
  values 23  006001  Longitude
  values 16  004001  Year
  values  8  002048  Satellite sensor type

END amsub
...```
# Observation formats in ARPEGE/ALADIN

- Elements inside square brackets are optional.
- Keywords must be written from the first column.
- BE CAREFUL: this file is case sensitive.

# BEGIN sensor
# a b c d
# [codage n1 desc1]
# ...
# [codage nn descn]
# [control n1 val1]
# ...
# [control nn valn]
# [offset n1 inc1]
# ...
# [offset nn1 incn]
# [values pos1 desc1]
# ...
# [values posn descn]
# END sensor

- Sensor must be in lowercase with name as defined in bator_dedbufr_mod.
- a is the number of 'codage' parameter defined.
- b is the number of 'control' parameter defined.
- c is the number of 'offset' parameter defined.
- d is the number of 'values' parameter defined.
- n1... nn = indice (integer).
- desc1... descn = BUFR descriptor FXY (must be unique).
- val1... valn = integer value used as reference for control.
- inc1... incn = integer value used as a jump.
- pos1... posn = index in the VALUES array (libemos).

- Codage is used to check the BUFR file structure with ktdlslst().
- Control is used to perform tests like number of channels...
- Offset defines a value to perform jump.
- Values are descriptors which will be used for decoding BUFR file.
Observation formats in ARPEGE/ALADIN

GRIB

- used for SEVIRI
- key routine bator_decodgrib_mod.F90
Outline of the talk

• Observation
  – overview
  – preprocessing
  – format

• Overview of the ODB software

• ODB applications
  – BATOR
  – ODBTOOLS
  – ODB text browsing (MANDALAY, odbviewer)
Observational DataBase (ODB) is tailor made database software developed at ECMWF to manage very large observational data volumes.

- **components:**
  - ODB/SQL language (definition of database and sql-compiler, flexible data layout definition & perform 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)

- **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
• 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) body: all information of one observed value ...)

• 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
Structure (hierarchy) is described in the Data Definition Layout file.

- ASCII file
- consists of uniquely named TABLEs
- tables are made up of uniquely named COLUMNS (or attributes)
  notation: column_name@table_name
- each COLUMN has a specific type
  - integer/real/string
  - packed
  - 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,
  column_name3 data_type3,
  ...
);
```
**ODB - relation between tables**

```
CREATE TABLE hdr AS (  
lat real,
lon real,
statid string,
obstype int
date YYYYMMDD  
body @LINK
);
```

```
CREATE TABLE body AS (  
varno pk5int,
obsvalue pk9real,
);
```

<table>
<thead>
<tr>
<th>lat</th>
<th>lon</th>
<th>statid</th>
<th>obstype</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.4</td>
<td>15.5</td>
<td>11518</td>
<td>1</td>
<td>20100913</td>
</tr>
</tbody>
</table>

...  

<table>
<thead>
<tr>
<th>varno</th>
<th>obsvalue</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>297.5</td>
</tr>
<tr>
<td>41</td>
<td>6.0</td>
</tr>
<tr>
<td>58</td>
<td>0.92</td>
</tr>
</tbody>
</table>

...  

```
<table>
<thead>
<tr>
<th>lat</th>
<th>lon</th>
<th>statid</th>
<th>obstype</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>50.4</td>
<td>14.5</td>
<td>11520</td>
<td>1</td>
<td>20100913</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>varno</th>
<th>obsvalue</th>
</tr>
</thead>
<tbody>
<tr>
<td>39</td>
<td>298.5</td>
</tr>
<tr>
<td>41</td>
<td>5.0</td>
</tr>
<tr>
<td>58</td>
<td>0.87</td>
</tr>
</tbody>
</table>
```

...  

```
<table>
<thead>
<tr>
<th>lat</th>
<th>lon</th>
<th>statid</th>
<th>obstype</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>52.4</td>
<td>16.5</td>
<td>11582</td>
<td>1</td>
<td>20100913</td>
</tr>
</tbody>
</table>
```
ODB - data retrieval

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 vis ODB-tools (odbviewer,...)

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 - compilation data flow

Data Definition Layout (DDL) file → ODB/SQL compiler → Intermediate C-code → Object code

Data query (SQL) files

Support libraries

Application program
ODB - miscellaneous items

Databases types:

- **ECMA** extended CMA (all tables needed for screening)
- **CCMA** compressed CMA (all tables needed for minimization)

Data Partitioning:

- to allow parallelism
  - TABLEs are divided horizontally into ”pools” between processors;
  - Pools are allocated to the MPI-tasks. By default, an MPI task cannot modify data on a pool that it does not own.
- number of pools is defined in the Fortran90 layer
- distribution can be done according to latitude bands, time-slot,

ODB I/O method:

- set via environment variable `$ODB_IO_METHOD`
- reflects the various ways ODB performs data access
- ODB currently supports 5 I/O methods, but
  - 1 - creates one file per every TABLE per data pool. Default for Météo France.
  - 4 - each similar TABLE-file for a number of consecutive pools (`$ODB_IO_GRP_SIZE`) are concatenated together into a single file, which cannot exceed `$ODB_IO_FILESIZE`
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 database metadata
- allow use MPI
- selected data can be asked to be "part-exchanged" across processors; but default data selection applies to the local pools only
- each query needs to be pre-compiled/linked with the main user program
- each cycle has its own ODB version!
- used 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
ODB in ARPEGE/ALADIN

ODB (ECMA or CCMA)

OPENDB → GETDB → PUTDB

"writing"

Reading

GETDB → PUTDB

CLOSEDB

Save

Subroutine #1 ...... Subroutine #N


**ODB in ARPEGE/ALADIN**

- **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`

  ```fortran
  HDR_LOOP: do jobs=1, NROWS_ROBHDR
    ROBHDR(jobs,MDBLAT) = <some_thing>
  enddo HDR_LOOP

  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
  ```

- **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
Outline of the talk

- Observation
  - overview
  - preprocessing
  - format

- Overview of the ODB software

- ODB applications
  - BATOR
  - ODBTOOLS
  - ODB text browsing (MANDALAY, odbviewer)
BATOR

program to create ODB database ECMA comprises

- conversion of observation into ODB format (from ASCII, BUFR or GRIB)
- assignment of observation errors eventually other information

see ECTERO(obstype, subtype, variable, level) in bator_init_mod.F90

ECTERO(NSYNOP,:,39,1)=1.4 ! SYNOP T2m
ECTERO(NSYNOP,1,41:42,1)=2.0 ! wind itsp=1
ECTERO(NSYNOP,2,41:42,1)=3.0 ! wind itsp=2 (ship)
...

- blacklisting
  - information defined in two ASCII files
    - -- LISTE_NOIRE_DIAP
to blacklist a whole set of observations
    - -- LIST_LOC
to blacklist some specific observations at some locations
  - no need for recompilation of BATOR executable
Example:
1 SYNOP 11 1 02045 03061996
5 TEMPMOBIL 37 58 AMDAR 28032002
6 PROFILER 34 4 70197 01062002
6 PROFILER 34 4 70197 0 PROF2 700 400 1 1 0
6 PROFILER 34 3 70197 0 PROF2 700 400 1 1 1 H06 H18

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Observation type (obstype@hdr)</td>
<td>i2</td>
</tr>
<tr>
<td>2</td>
<td>Observation name</td>
<td>a10</td>
</tr>
<tr>
<td>3</td>
<td>Observation codetype (codetype@hdr)</td>
<td>i3</td>
</tr>
<tr>
<td>4</td>
<td>Parameter ID (varno@body)</td>
<td>i3</td>
</tr>
<tr>
<td>5</td>
<td>Station ID (statid@hdr)</td>
<td>a8</td>
</tr>
<tr>
<td>6</td>
<td>The starting date of blacklisting yyyyymmdd</td>
<td>a8</td>
</tr>
<tr>
<td>7</td>
<td>Optional layers blacklisting keyword for PROFn</td>
<td>a180</td>
</tr>
</tbody>
</table>

PROFn P1a P2 ... Pn-1 I1 I2 ... In-1

- n can be at most 9 indicating the involved layers
- the Pi values specify the bottom and top levels of pressure layers (in hPa). The first layer is always \([1000, P1]\)
- the Ii values indicate if blacklisting should be applied (=1) or not (=0) to the given layer. The Hxx keyword specifies the analysis hour that should be blacklisted e.g. H00 or H06 etc
Particularities - the blacklisting of certain parameters involves the automatic blacklisting of other parameter summarized in the table below:

<table>
<thead>
<tr>
<th>obstype</th>
<th>specified parameter</th>
<th>blacklisted parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYNOP</td>
<td>39 (t2)</td>
<td>39 (t2), 58 (rh2), 7 (q)</td>
</tr>
<tr>
<td>SYNOP</td>
<td>58 (rh2)</td>
<td>58 (rh2), 7 (q)</td>
</tr>
<tr>
<td>TEMP</td>
<td>1 (z)</td>
<td>1 (z), 29 (rh), 2 (t), 59 (td), 7 (q)</td>
</tr>
<tr>
<td>TEMP</td>
<td>2 (t)</td>
<td>2 (t), 29 (rh), 7 (q)</td>
</tr>
<tr>
<td>TEMP</td>
<td>29 (rh)</td>
<td>29 (rh), 7 (q)</td>
</tr>
</tbody>
</table>
### Example:

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type of action: N for blacklisted</td>
<td>a1</td>
</tr>
<tr>
<td>2</td>
<td>The observation type (obsytype@hdr)</td>
<td>i3</td>
</tr>
<tr>
<td>3</td>
<td>The observation code-type (codetype@hdr)</td>
<td>i4</td>
</tr>
<tr>
<td>4</td>
<td>The satellite ID with leading zeros (satid@sat)</td>
<td>a9</td>
</tr>
<tr>
<td>5</td>
<td>The centre that produced the satellite data</td>
<td>i4</td>
</tr>
<tr>
<td>6</td>
<td>The parameter ID (varno@body) or the satellite sensor ID (sensor@hdr)</td>
<td>i4</td>
</tr>
<tr>
<td>7</td>
<td>Optional keywords of ZONx4, TOVSn, PPPPn, PROFn</td>
<td></td>
</tr>
</tbody>
</table>

```
N 2 145 29
N 3 88 052
N 7 210 206 3 TOVS2 6 11
N 7 210 206 0 TOVS6 4 5 6 7 8 9
N 7 210 206 0 TOVS6 10 11 12 13 14 15
N 7 210 207 0 TOVS6 4 5 6 7 8 9
N 7 210 207 0 TOVS6 10 11 12 13 14 15
N 7 210 208 0 TOVS6 4 5 6 7 8 9
N 7 210 208 0 TOVS6 10 11 12 13 14 15
N 7 210 208 3 TOVS6 4 5 6 7 8 9
N 7 210 208 3 TOVS6 10 11 12 13 14 15
...```
BATOR - blacklisting via LISTE_LOC

TOVSn C1 C2 ... Cn
- can be applied to ATOVS radiances
- n can be at most 9 indicating the involved channels
- the Ci values specify the channels to be blacklisted

PPPPn P1 P2 ... Pn
- can be applied to blacklist different pressure levels
- n can be at most 9 indicating the involved levels
- the Pi values specify the pressure levels (in hPa) to be blacklisted

PROFn P1a P2 ... Pn-1 I1 I2 ... In-1
- n can be at most 9 indicating the involved layers
- the Pi values specify the bottom and top levels of pressure layers (in hPa).
  The first layer is always [1000,P1]
- the Ii values indicate if blacklisting should be applied (=1) or not (=0) to
  the given layer.

ZONx4 latmin latmax lonmin lonmax
- can be applied to SATOB/GEOWIND data
- if x=B then the pixels with lat < latmin or lat > latmax or lon < lonmin or
  lon > lonmax will be blacklisted
- if x=C then the pixels with lat < latmin or lat > latmax or (lon > lonmin
  and lon < lonmax) will be blacklisted.
BATOR - I/O summary

Inputs:

- setting of environmental variables
  - ODB_IO_METHOD =1
  - ODB_CMA=ECMA
  - IOASSIGN= path to IOASSIGN file
  - ODB_SRCPATH_ECMA = the location of ODB sub-bases’ description files
  - ODB_DATAPATH_ECMA = the location of ODB sub-bases' data files
  - BATOR_NBPOOL = the number of the pools in the resulting ODB sub-bases
  - BATOR_NBSLOT = the number of timeslots (1 for 3D-VAR)
  - BATOR_LAMFLAG = 0/1
  - ODB_ANALYSIS_DATE = the date (yyyymmddd) of the analysis
  - ODB_ANALYSIS_TIME = the time (hh0000) of the analysis

- file ficdate containing the time-slot definition, e.g. for 3DVAR on 2010071800
  20100717210000
  20100718030000

- file refdata describing the input files
  conv OBSOUL conv 20100718 0
  tovamsua BUFR amsua 20100718 0
  sev GRIB sev 20100718 0
BATOR - I/O summary

Inputs:

- file LISTE_NOIRE_DIAPI (optional) for blacklisting
- file LISTE_LOC (optional) for blacklisting
- NAM_lamflag namelist for performing LAMFLAG filtering (if BATOR_LAMFLAG=1)
- namelist_rgb namelist for reading the SEVIRI GRIB data
- param.cfg a configuration file to decode bufr-files
- NAMELIST optional namelist for BATOR
- observation inputs: in OBSOUL, BUFR or GRIB format
  OBSOUL.conv, bufr.tovamsua, grib.sev

Execution: mpirun -np 1 ./BATOR

Output:
ODB sub-base (the suffix is specified in file refdata)
ECMA.base (e.g. ECMA.conv, ECMA.amsua, ..)

These ODB sub-bases must be merged into a full ECMA ODB by programme SHUFFLE (ODBTOOLS)
program to perform a geographical and an observational selection

- specific to ALADIN, if skipped the screening aborts
- formerly a separate program, but from CY30 integrated in BATOR
- invoked via environment `BATOR_LAMFLAG = 1`
- requires a specific namelist `NAM_lamflag` that defines:
  
  - **the limits of the domain and reduction if C+I zone** `&NAMFGEOM`  
    
    ELAT0 ELON ELATC ELONC ELAT1 ELON1 REDZONE REDZONE_N REDZONE_W ...  
    EDELX EDELY NDLUN NDGUN NDLUX NDGUX  
  
  - **types of observations to select** `&NAMFOBS`  
    LSYNOP LSATOB LTEMP LSATEM ...
program to perform various databases 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:

- ODB_IO_METHOD =1
- ODB_CMA = database type definition
- IOASSIGN = path to IOASSIGN file - the directory structure of the database
- ODB_SRCPATH_ECMA = the location of ODB sub-bases’ description files
- ODB_DATAPATH_ECMA = the location of ODB sub-bases’ data files

Examples:

- ECMA -> ECMA update (merge virtual ODB base - ”mergeodb”)
- ECMASCR -> ECMA translation (load balanced, 002 database)
- ECMA -> CCMA translation (load balanced, active data, 131 database)
- CCMA -> ECMA update (”matchup” )
ECMA –> ECMA

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:
  - ECMA.dd
  - ECMA.sch
  - ECMA.poolmask
  - ECMA.IOASSIGN

- ”merge” comprisis
  - creation of IOASSIGN file via `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, ...`

```bash
mpirun -np 1 ./shuffle -iECMA -oECMA -atotal_n_pools -b1
```
Minimization requires CCMA database (with only active observations)

- creation of IOASSIGN file via `create_ioassign`
  
  ```
  create_ioassign -lCCMA -n$NPROC
  ```

- a shuffle run - (creation of CCMA database)
  
  ```
  mpirun -np 1 ./shuffle -iECMA -oCCMA -b1 -a$NPROC -B$YYYYMMDDNT
  ```

# ODB settings for shuffle

```bash
export SWAPP_ODB_IOASSIGN=$WD/ioassign
export ODB_SRCPATH_ECMA=$WD/ECMA
export ODB_DATAPATH_ECMA=$WD/ECMA
export ODB_SRCPATH_CCMA=$WD/CCMA
export ODB_DATAPATH_CCMA=$WD/CCMA
export ODB_CCMA_CREATE_POOLMASK=1
export ODB_CCMA_POOLMASK_FILE=$WD/CCMA/CCMA.poolmask
```

# create output database directory and IOASSIGN file

```bash
mkdir CCMA
./create_ioassign -lCCMA -n$NPROC
cat IOASSIGN >> ECMA/IOASSIGN
cp ECMA/IOASSIGN CCMA/IOASSIGN
cp ECMA/IOASSIGN IOASSIGN
```

# run shuffle
ODB text browsing

Viewing the content of ODB

- odbviewer - "dynamic" retrieval based
  - compilation is done on the fly
  - available in an ODB-standalone package only

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

- mandaodb - "static" retrieval based
  - retrieval are based on predefined and user defined views
  - 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 ./MANDLAY CMAFILE
Thank You for Your attention.