Preprocessing and format of observations for the data assimilation

Alena Trojáková

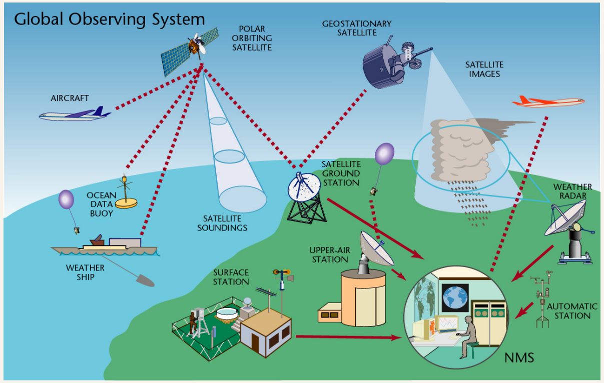
alena.trojakova@chmi.cz

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,..
    ($p, T2m, RH2m, v10m, v10m, RR, SST, ..$)
  - Upper-air & Aircraft - TEMP,PROFILER,PILOT,AMDAR,SATOB,..
    (u, v, T, q, ϕ)
- satellite
  - AIRS,AMSU-A/B,ASCAT,HIRS,IASI,MHS,SEVIRI,..
    (Tb)
- other platforms (Doppler radar, solar radiation observations, ..)



Observation types in ARPEGE/ALADIN

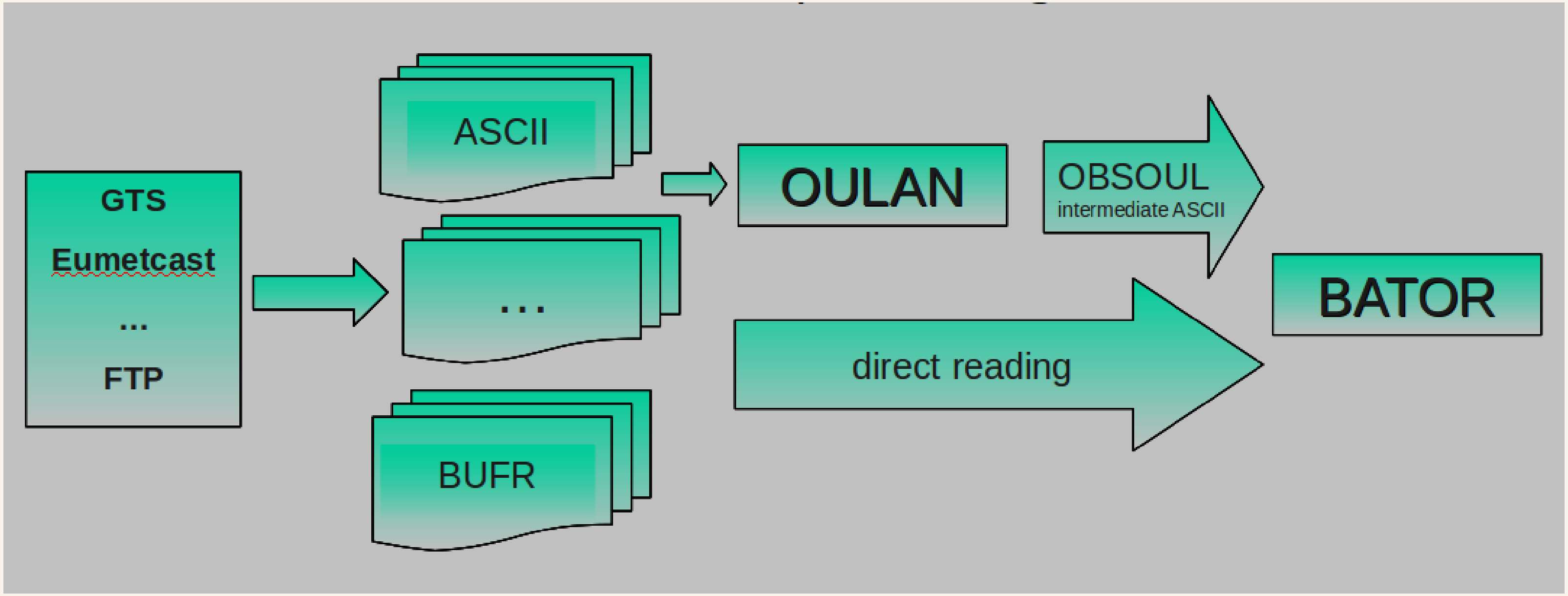
Variable	Value	Observation type
NSYNOP	1	SYNOP, SYNOP_SHIP, SYNOR
NAIREP	2	AIREP, AMDAR, ACAR, CODAR, COLBA
NSATOB	3	SATOB
NDRIBU	4	DRIBU, DRIFTER, BUOY, BATHY, TESAC, ERS1
NTEMP	5	TEMP, TEMP-SHIP, TEMP_DROP, ROCOB
NPILOT	6	PILOT, PILOT-SHIP, PILOTMOBIL, wind profiler
NSATEM	7	SATEM, TOVS
NPAOB	8	PAOB
NSCATT	9	scaterrometer(ERS)
NLIMB	10	GPS
NRADAR	13	radar

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

	Value	variable name [unit]
NVNUMB(1)	3	u-wind component [m/s]
NVNUMB(2)	4	v wind component [m/s]
NVNUMB(3)	1	geopotentiel [J/kg]
...		
NVNUMB(96)	189	N2O

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

Observation preprocessing

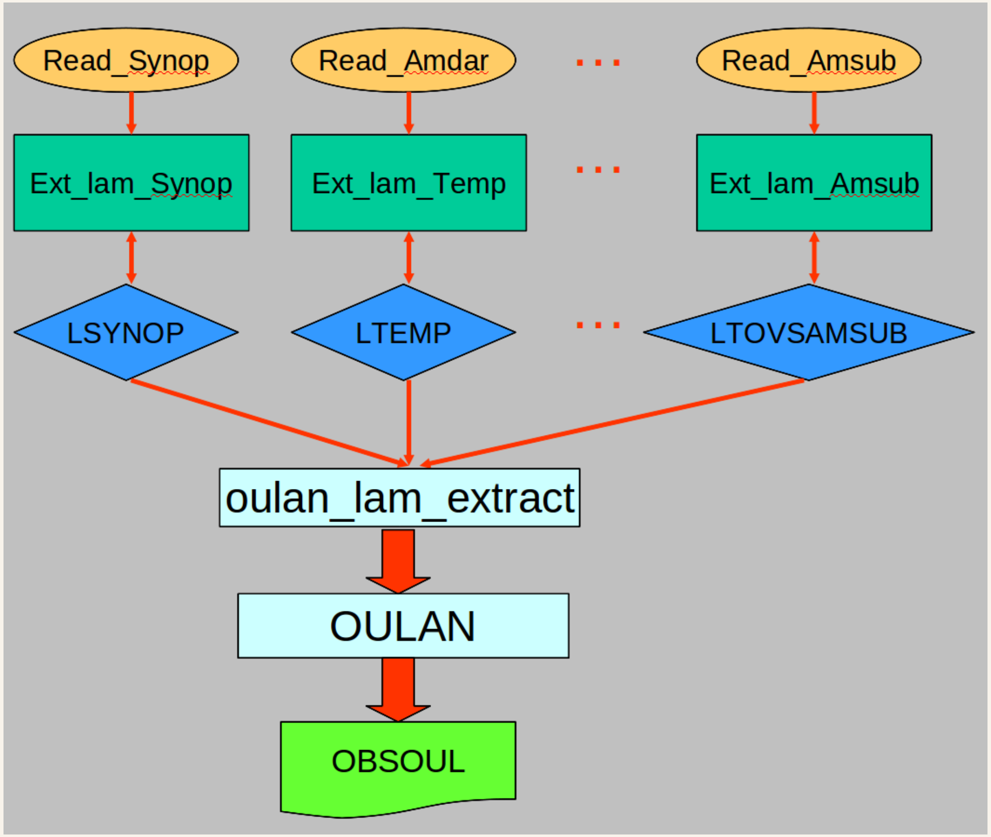


- preparation for use in NWP (data assimilation, verification, nowcasting, ...)
  - reception and storage
  - decoding and/or format conversion
  - local pre-treatment (generation of necessary parameters for advanced data (SEVIRI), initialization of various flags)
  - very basic quality control
  - conversion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

OULAN

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



Observation formats in ARPEGE/ALADIN

OBSOUL (ASCII)

- simple format
- suitable for testing of new observation types

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

date: `yyyymmdd`

time: `hhmmss`

record: `n header body1 ... bodyk`

	Header Description	Type
1	observation type (obstype@hdr)	integer
2	observation code	integer
3	latitude	real
4	longitude	real
5	station/satellite identification	character
6	date *yyyymmdd*	integer
7	time *hh*	integer
8	altitude	real
9	number of parameters (= number of bodies)	integer
10	observation quality flags	integer
11	site dependant	integer

Observation formats in ARPEGE/ALADIN

	Body Description	Type
1	type of parameter (varid@body)	integer
2	first vertical coordinate	real
3	second vertical coordinate (if necessary of other)	real
4	observed or measured parameter	real
5	parameter quality flag	integer

```
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 Univeral Form for data Representatation) - has been designed to achive efficent 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
...
```

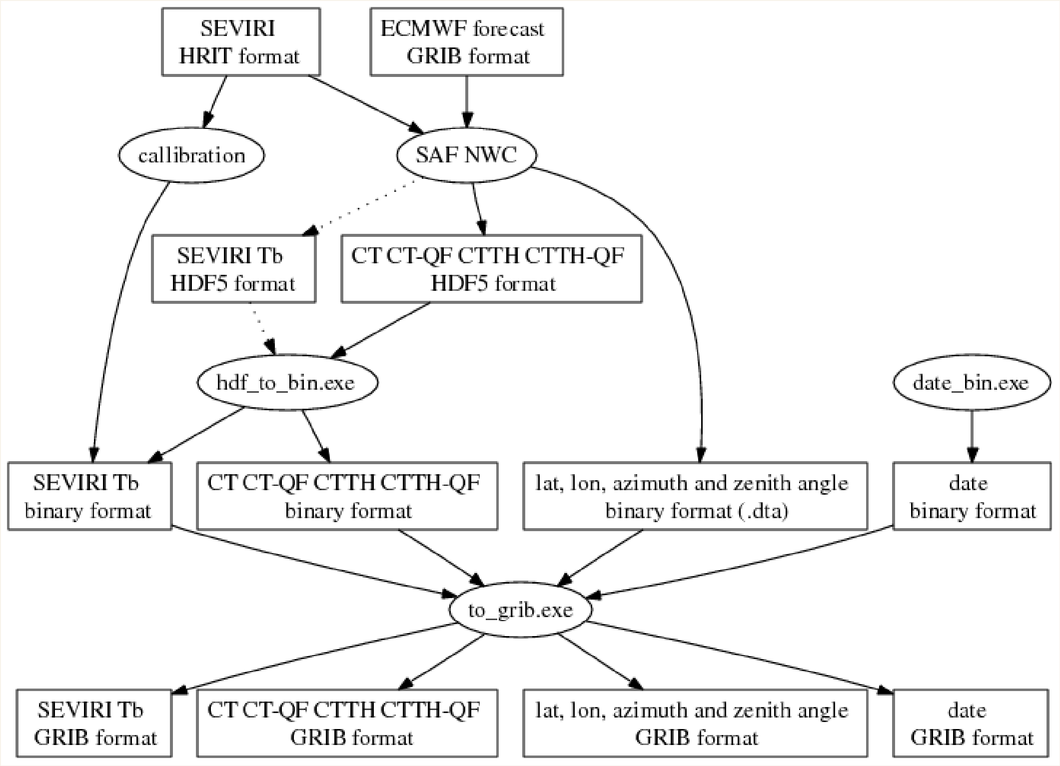
Observation formats in ARPEGE/ALADIN

```
# 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 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 ktdlst(),
# 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)

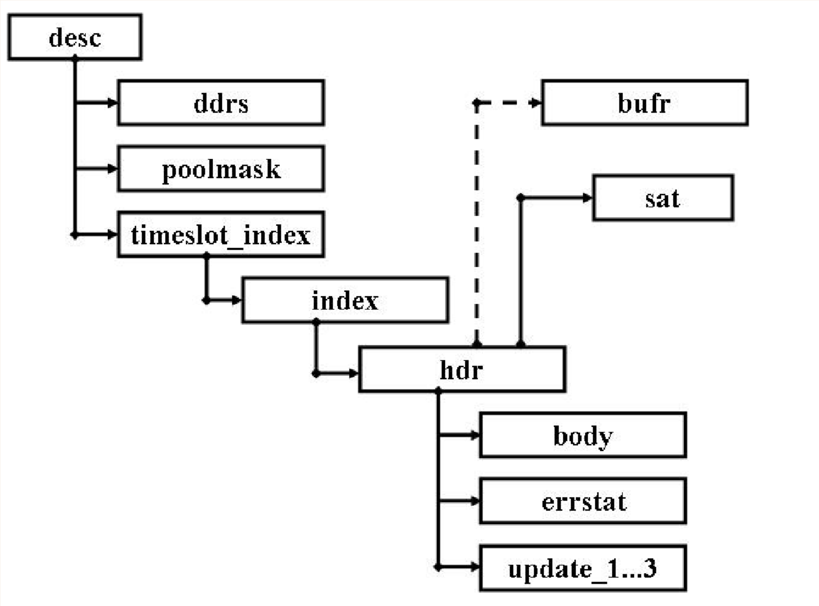
ODB - general information

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

ODB - structure

- structure:
  - basic building blocks called table (can be seen as a matrice (2D-array)) with a number of rows and columns containing numberical 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

ODB - Data Definition Layout (DDL)

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,
);
```

lat	lon	statid	obstype	date
50.4	15.5	11518	1	20100913

...

lat	lon	statid	obstype	date
50.4	14.5	11520	1	20100913

lat	lon	statid	obstype	date
52.4	16.5	11582	1	20100913

...

varno	obsvalue
39	297.5
41	6.0
58	0.92

...

body@LINK → (offset=j, length=3)

varno	obsvalue
39	298.5
41	5.0
58	0.87

...

ODB - data retrieval

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

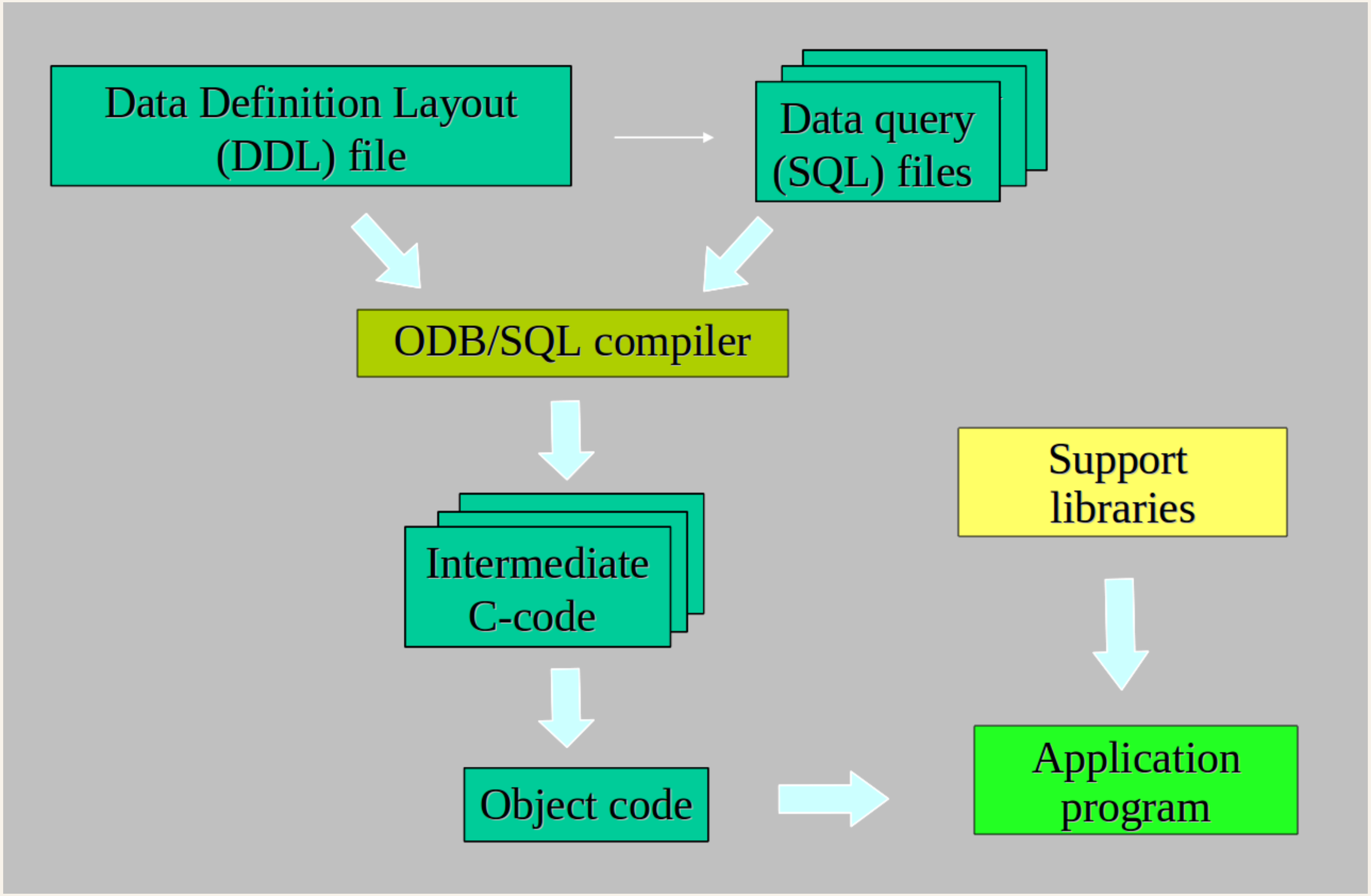
> [CREATE VIEW view_name AS]
> SELECT [DISTINCT] column_name (s)
> FROM table(s)
> WHERE cond ORDERBY sort_column_name(s) [ASC/DESC]

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



ODB - miscelaneous items

Databases types:

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

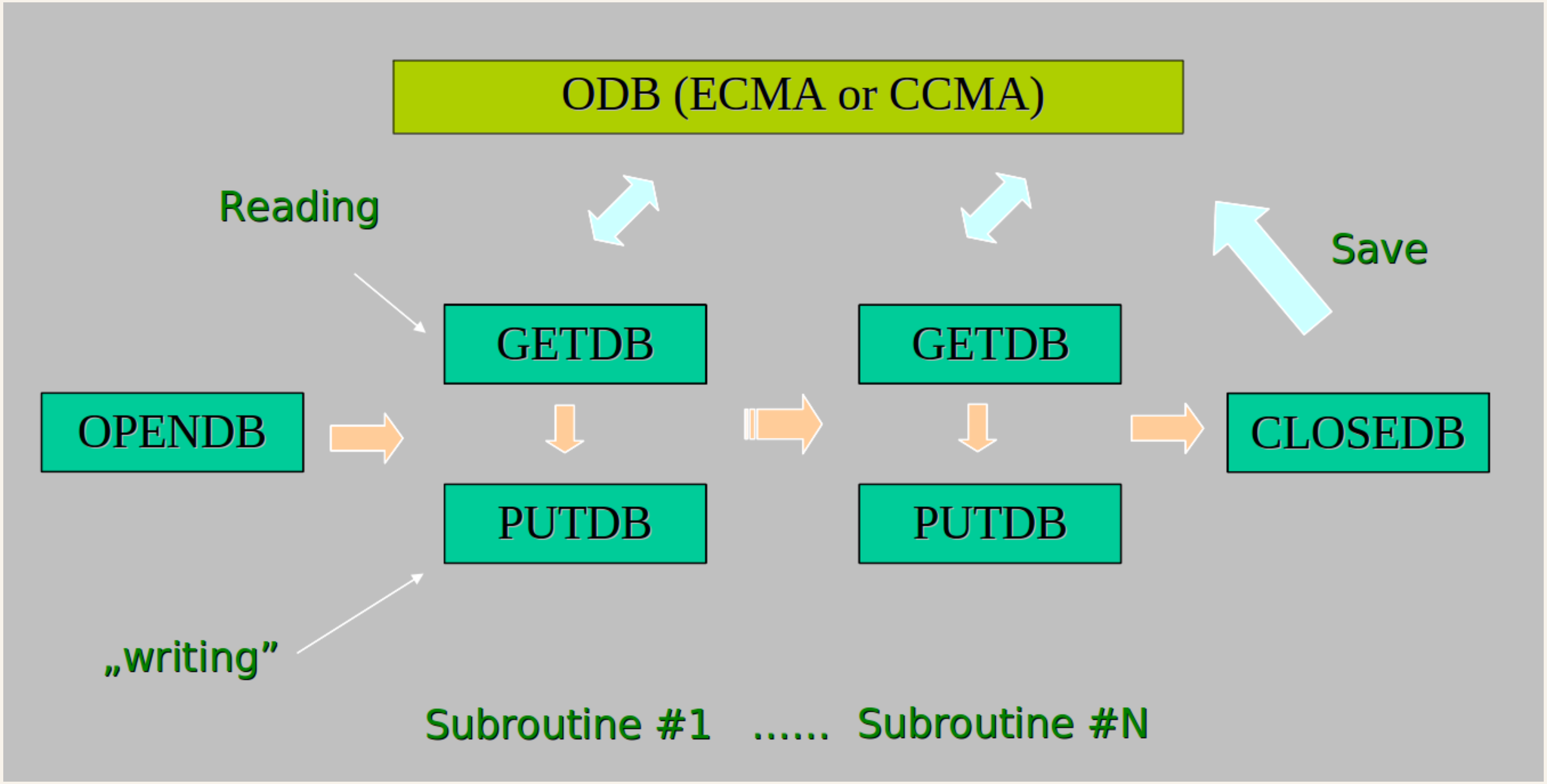
ODB I/O method:

- set via enviromnment variable $ODB_IO_METHOD
- reflects the varyous ways ODB performs data access
- ODB currently support 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_GRPSIZE) 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" acroess processors; bu default data selection applies to the local pools only
- each query need 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 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

```
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

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

BATOR - blacklisting via LISTE_NOIRE_DIAP

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
```

Column	Description	Format
1	Observation type (obstype@hdr)	i2
2	Observation name	a10
3	Observation codetype (codetype@hdr)	i3
4	Parameter ID (varno@body)	i3
5	Station ID (statid@hdr)	a8
6	The starting date of blacklisting `yyyymmdd`	a8
7	Optional layers blacklisting keyword for PROFn	a180

`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**

BATOR - blacklisting via LISTE_NOIRE_DIAP

Particularities - the blacklisting of certain parameters involves the automatic blacklisting of other parameter summarized in the table below:

obstype	specified parameter	blacklisted parameters
SYNOP	39 (t2)	39 (t2), 58 (rh2), 7 (q)
SYNOP	58 (rh2)	58 (rh2), 7 (q)
TEMP	1 (z)	1 (z), 29 (rh), 2 (t), 59 (td), 7 (q)
TEMP	2 (t)	2 (t), 29 (rh), 7 (q)
TEMP	29 (rh)	29 (rh), 7 (q)

BATOR - blacklisting via LISTE_LOC

Example:

```
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
...
```

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

BATOR - blacklisting via LISTE_LOC

`TOVSn C1 C2 ... Cn`

- can be aplied 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 aplied 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 (yyyymmdd) 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_DIAP** *(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

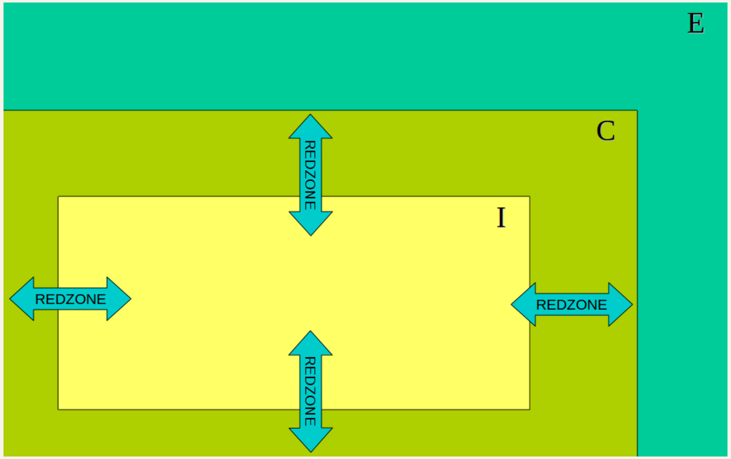
Output:
ODB sub-base (the suffix is specified in file refdata)
ECMA.base (e.g. ECMA.conv, ECMA.amsua, ..)

These ODB sub-bases must be merged into a full ECMA ODB by programme SHUFFLE (ODBTOOLS)

LAMFLAG

program to perform a geographical and an observational selection

- specific to ALADIN, if skipped the screening aborts
- formerly a separate program, but from CY30 intergrated 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`
    `ELATO 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 ...`



ODBTOOLS

program to perform various databses shuffles

- data repartition
- change of the number of the pool
- timeslot and time-window definition
- data selection

Execution is controlled by a set of environemntal 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*

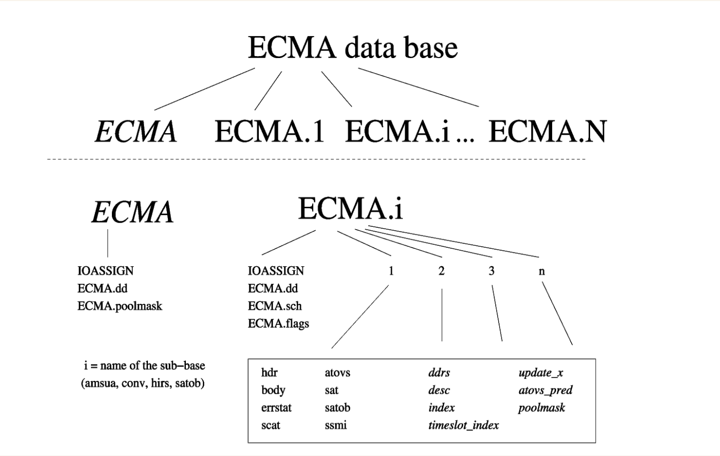
Examples:

- ECMA $->$ ECMA update (merge virtual ODB base - "mergeodb")
- ECMASCR $->$ ECMA translation (load balanced, 002 database)
- ECMA $->$ CCMA translation (load balanced, adtive 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, ...*

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

ECMA − > CCMA

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
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
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
```

The End

Thank You for Your attention.