Norwegian
Meteorological
Institute

Data assimilation Monitoring/Validation/Diagnostic

Roger Randriamampianina

21.09.18

DasKit working days, 19-21 September 2018, Bucharest

outline

- Short introduction of data assimilation;

- Observation post-processing;

- DA monitoring – Obsmon tool.

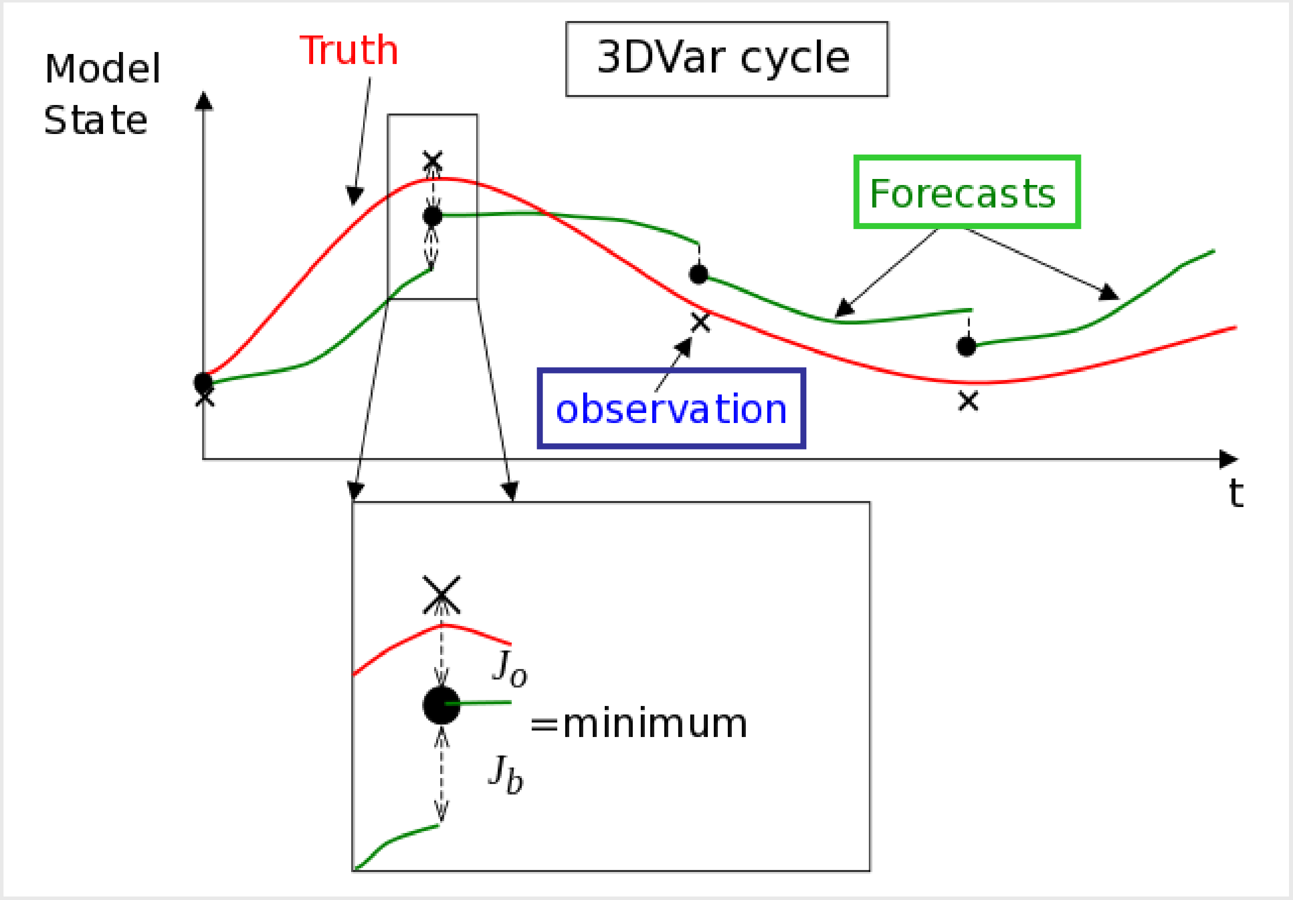
- DA diagnostic using DFS, ObsTool and MTEN;

Data assimilation - 1

Data assimilation is the process by which observations are incorporated into a computer model of a real system. Applications of data assimilation arise in many fields of geosciences, perhaps most importantly in weather forecasting and hydrology. Data assimilation proceeds by analysis cycles. In each analysis cycle, observations of the current (and possibly past) state of a system are combined with the results from a numerical model (the forecast) to produce an analysis, which is considered as 'the best' estimate of the current state of the system. This is called the analysis step. Essentially, the analysis step tries to balance the uncertainty in the data and in the forecast. The model is then advanced in time and its result becomes the forecast in the next analysis cycle.

Source: wikipedia

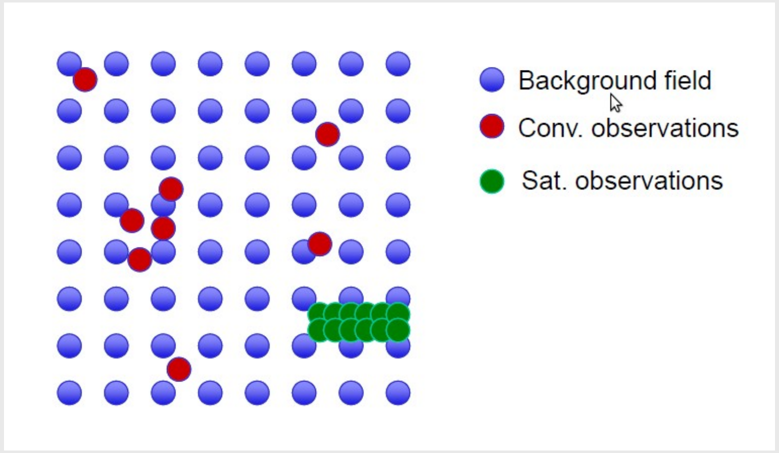
Data assimilation - 2



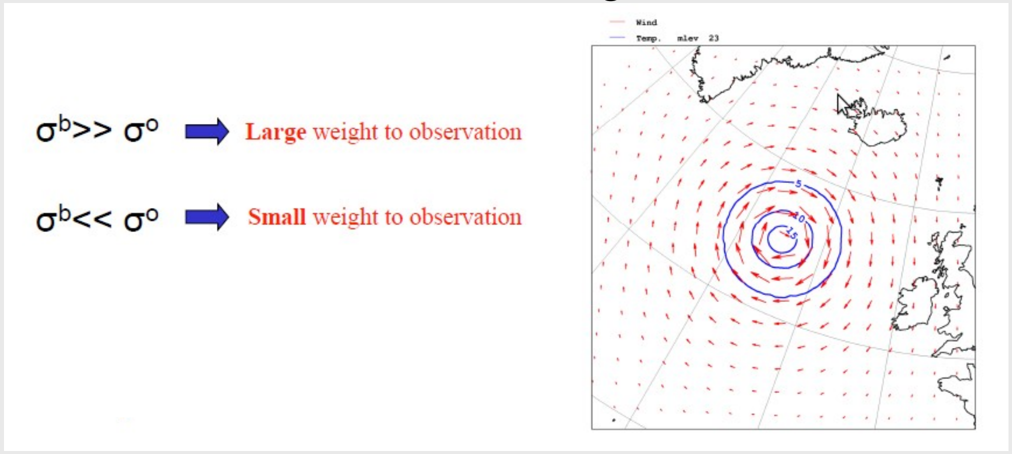
Courtesy of T. Montmerle

Data assimilation - 3

Observations and background



Observation and background errors


$\sigma^b >> \sigma^o$ ➡ Large weight to observation
$\sigma^b << \sigma^o$ ➡ Small weight to observation
Wind
Temp, mlev 23

Influence of single observation

Optimum interpolation – using the best linear unbiased estimation: **BLUE**

$$\mathbf{x}_a = \mathbf{x}_b + \mathbf{W}[\mathbf{y}_o - H(\mathbf{x}_b)]$$

$$\mathbf{W} = \mathbf{B}\mathbf{H}^T(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1}$$

X_a – analysis; X_b – background; y_0 – observation
H/H – observation operator; R – observation error covariance; B – background error covariance.

The background **error covariance, B**, and the **observation error covariance, R**, are assumed to be known.
We assume that **the observation and background errors are uncorrelated**:

Courtesy of Lindskog

Data assimilation monitoring

Optimum interpolation – using the best linear unbiased estimation: **BLUE**

$$\mathbf{x}_a = \mathbf{x}_b + \mathbf{W}[\mathbf{y}_o - H(\mathbf{x}_b)]$$

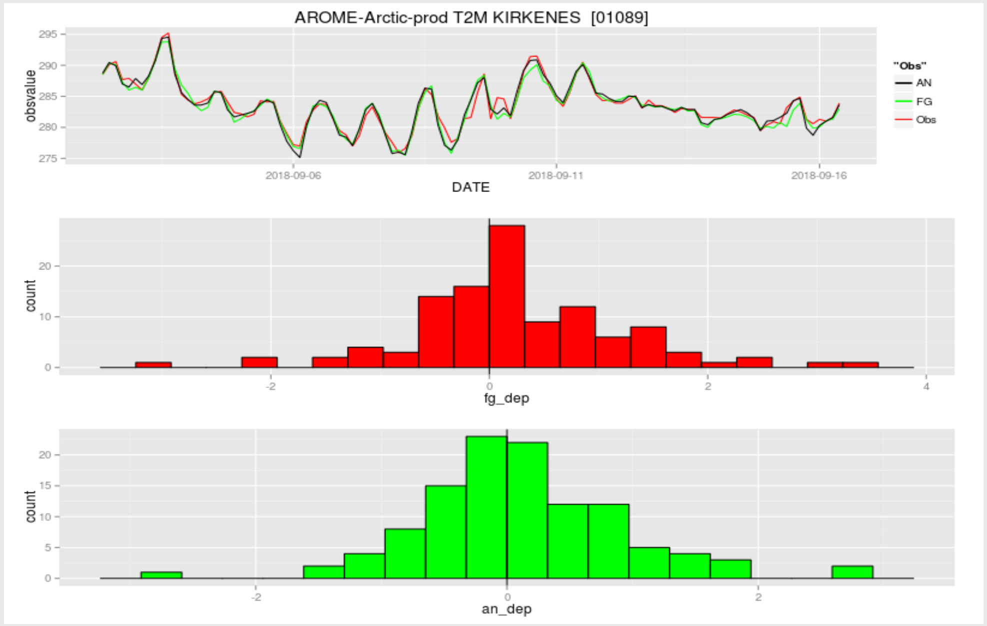
$$\mathbf{W} = \mathbf{B}\mathbf{H}^T(\mathbf{R} + \mathbf{H}\mathbf{B}\mathbf{H}^T)^{-1}$$

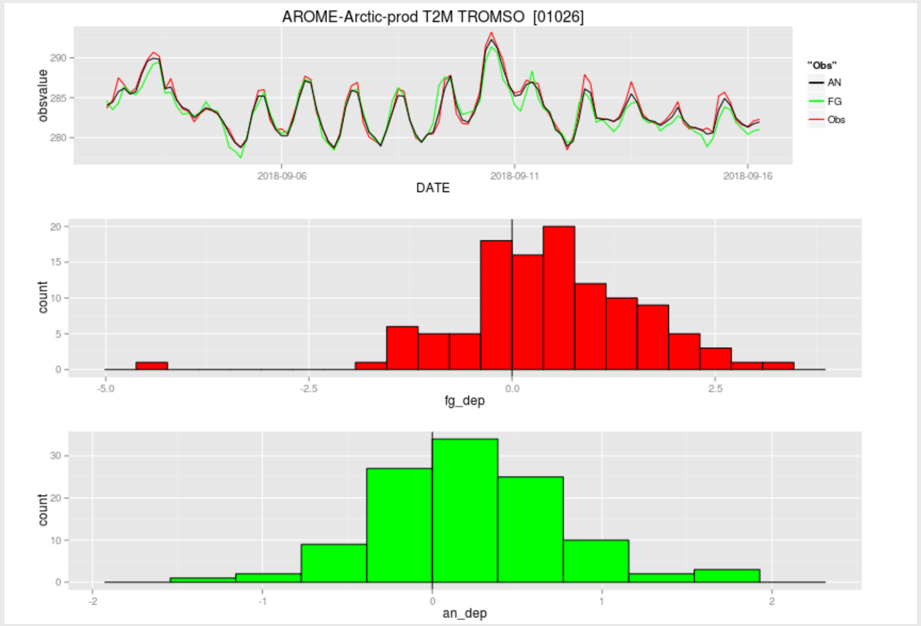
X_a – analysis; X_b – background; y_0 – observation *H*/H – observation operator; R – observation error covariance; B – background error covariance.

We usually choose the following parameters/outputs for monitoring:

- fg_depar (in ODB): $y_0 - H(x_b)$: innovation or first-guess departure – in observation space
- an_depart (in ODB): $y_0 - H(x_a)$: analysis departure
- Observation usage status (spatial and temporal)

Example of temporal efficiency





Obsmon monitoring tool

Obsmon consists of two components:

– Observation post-processing and a shiny web interface.

I sent an instruction on how to install the obsmon shiny interface.

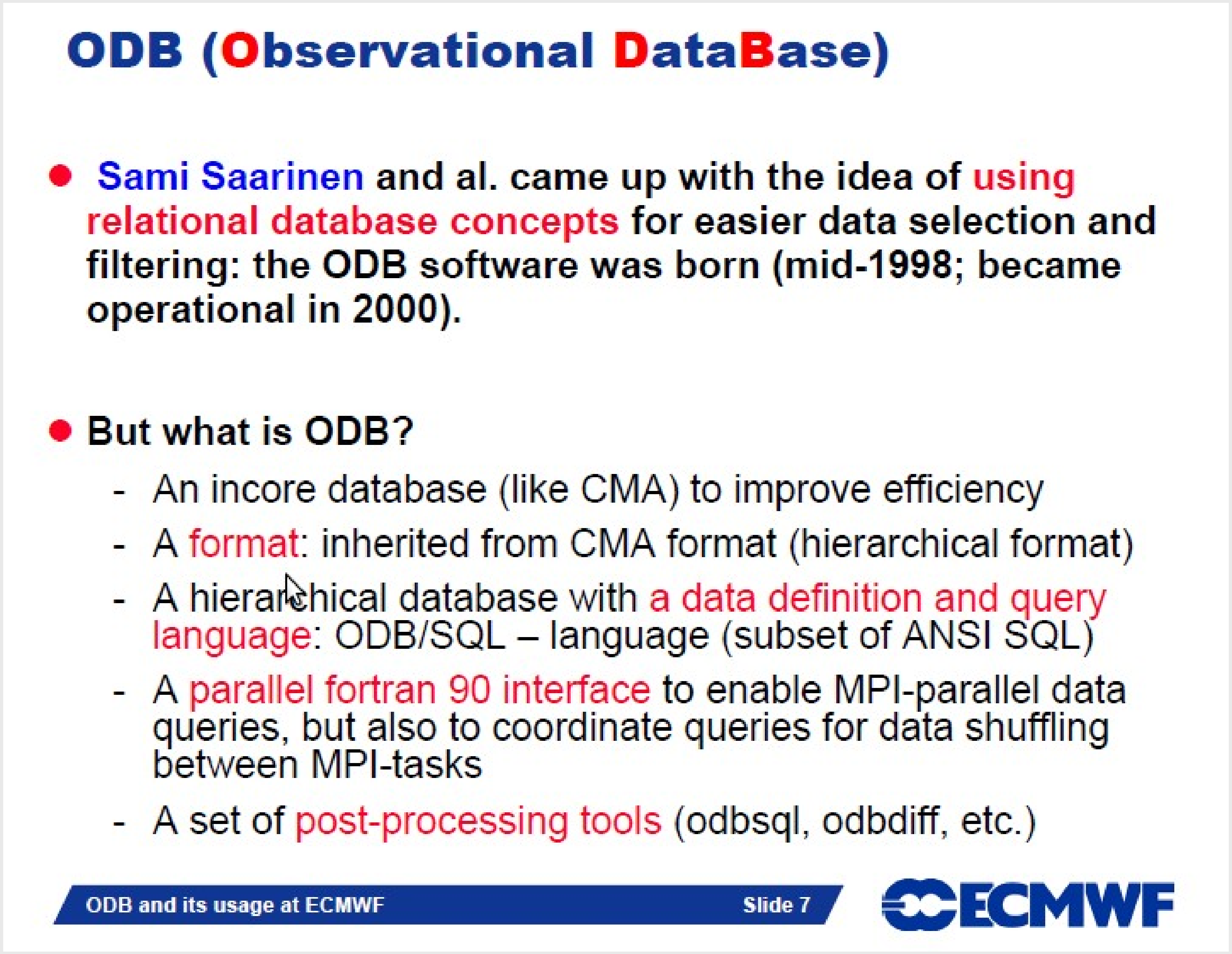
Similarly, I also sent how to install the obsmon post-processing part.

We monitor the functionality of the analysis system by analysing the content of ODB database: ECMA – input and output of the screening; CCMA – input and outpu of minimisation, so the objective analysis process.

Focusing on surface DA, we have only ECMA, which is updated by Canari – the OI scheme.

To do the monitoring, we need to know well the ODB.

Short introduction to ODB, if needed



ODB (Observational DataBase)

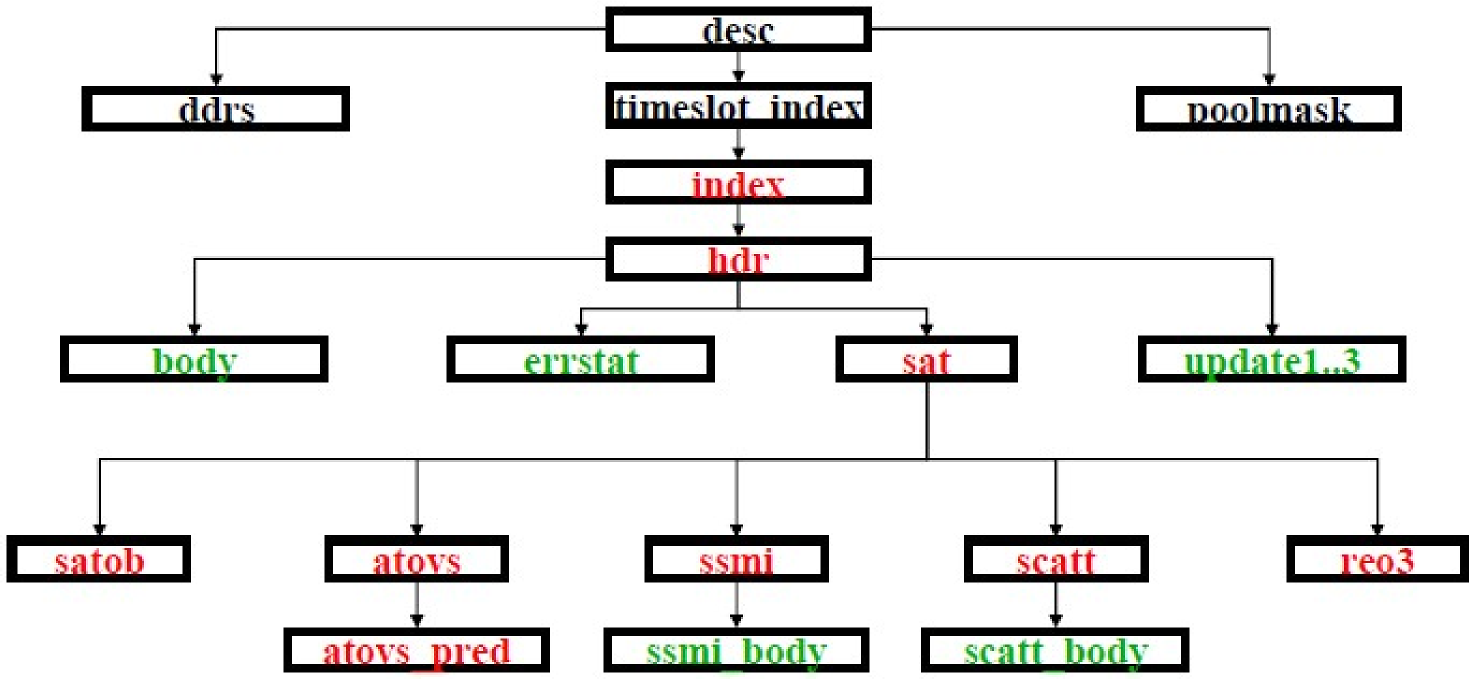
- Sami Saarinen and al. came up with the idea of using relational database concepts for easier data selection and filtering: the ODB software was born (mid-1998; became operational in 2000).

- But what is ODB?
  - An incore database (like CMA) to improve efficiency
  - A format: inherited from CMA format (hierarchical format)
  - A hierarchical database with a data definition and query language: ODB/SQL – language (subset of ANSI SQL)
  - A parallel fortran 90 interface to enable MPI-parallel data queries, but also to coordinate queries for data shuffling between MPI-tasks
  - A set of post-processing tools (odbsql, odbdiff, etc.)

ECMWF

Reference: Anne Fouilloux (ECMWF)

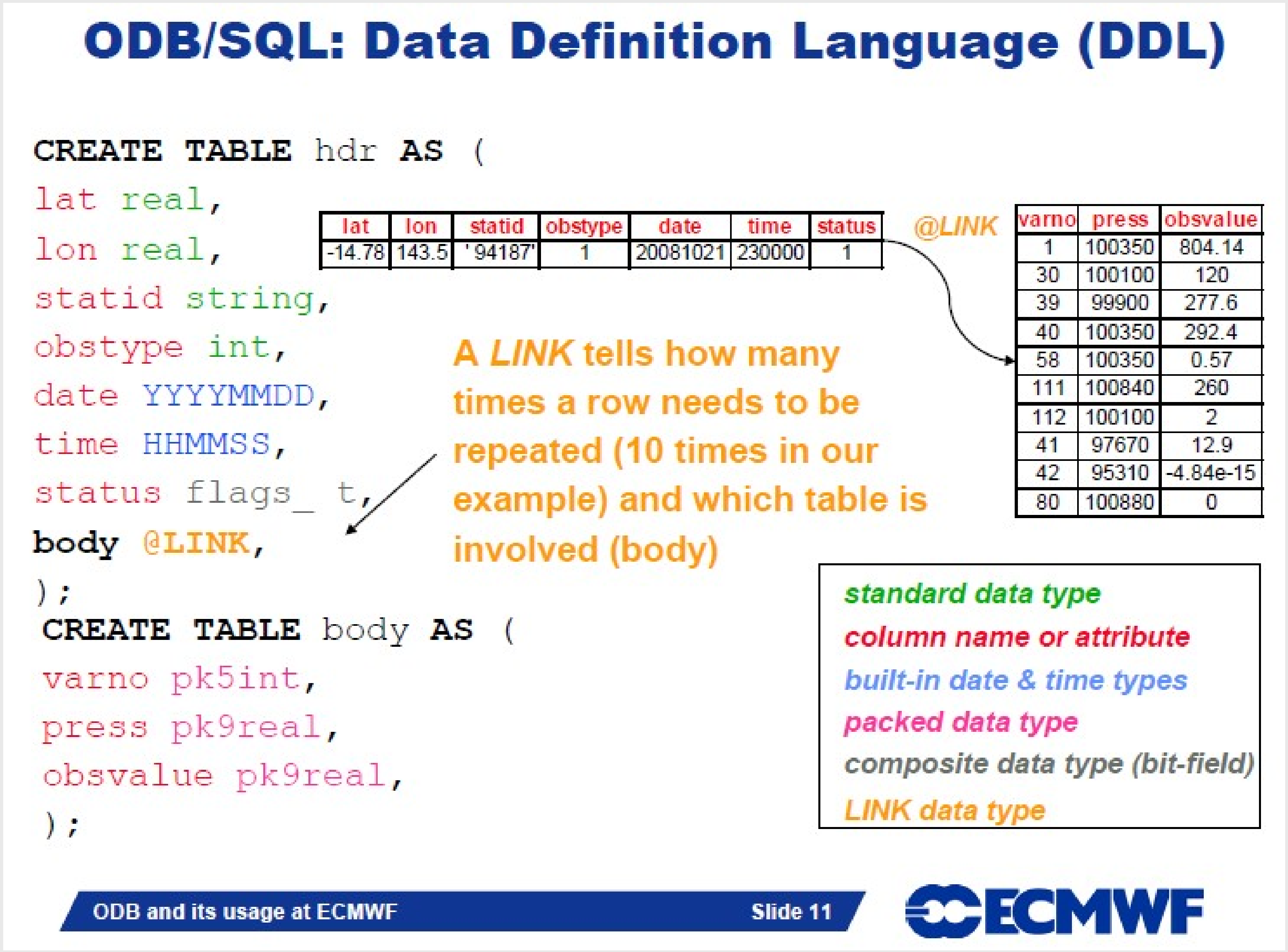
ODB hierarchical data model – ECMWF layout



- *ssmi_body, scatt_body* tables are similar to body table

ECMWF

Reference: Anne Fouilloux (ECMWF)



ODB/SQL: Data Definition Language (DDL)

```
CREATE TABLE hdr AS (
lat real,
lon real,
statid string,
obstype int,
date YYYYMMDD,
time HHMMSS,
status flags_ t,
body @LINK,
);
 CREATE TABLE body AS (
 varno pk5int,
 press pk9real,
 obsvalue pk9real,
 );
```

lat	lon	statid	obstype	date	time	status
-14.78	143.5	'94187'	1	20081021	230000	1

@LINK

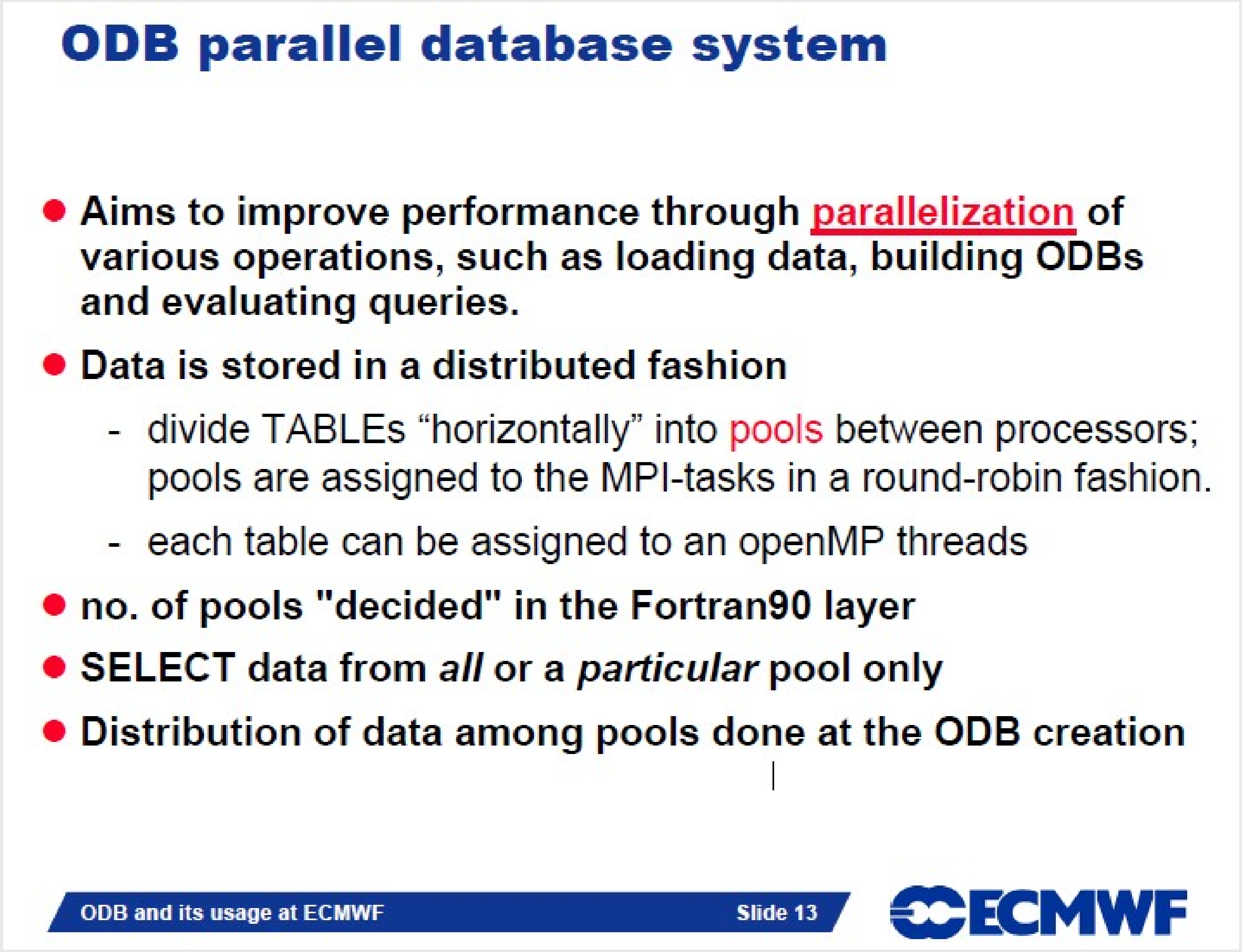
varno	press	obsvalue
1	100350	804.14
30	100100	120
39	99900	277.6
40	100350	292.4
58	100350	0.57
111	100840	260
112	100100	2
41	97670	12.9
42	95310	-4.84e-15
80	100880	0

A *LINK* tells how many times a row needs to be repeated (10 times in our example) and which table is involved (body)

standard data type
column name or attribute
built-in date & time types
packed data type
composite data type (bit-field)
LINK data type

ECMWF

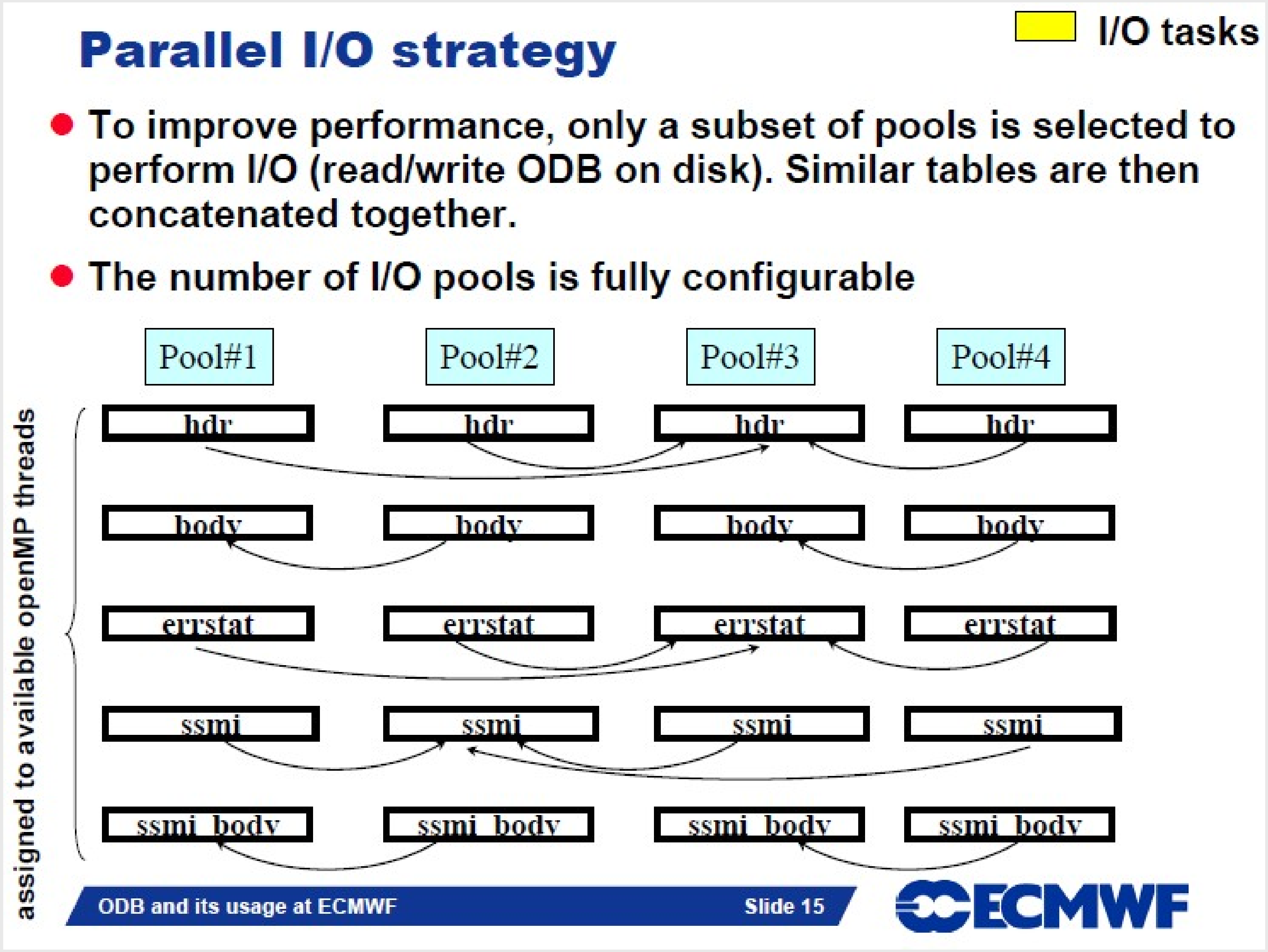
Reference: Anne Fouilloux (ECMWF)



ODB parallel database system

- Aims to improve performance through parallelization of various operations, such as loading data, building ODBs and evaluating queries.
- Data is stored in a distributed fashion
  - divide TABLEs “horizontally” into pools between processors; pools are assigned to the MPI-tasks in a round-robin fashion.
  - each table can be assigned to an openMP threads
- no. of pools "decided" in the Fortran90 layer
- SELECT data from *all* or a *particular* pool only
- Distribution of data among pools done at the ODB creation

Reference: Anne Fouilloux (ECMWF)



Parallel I/O strategy

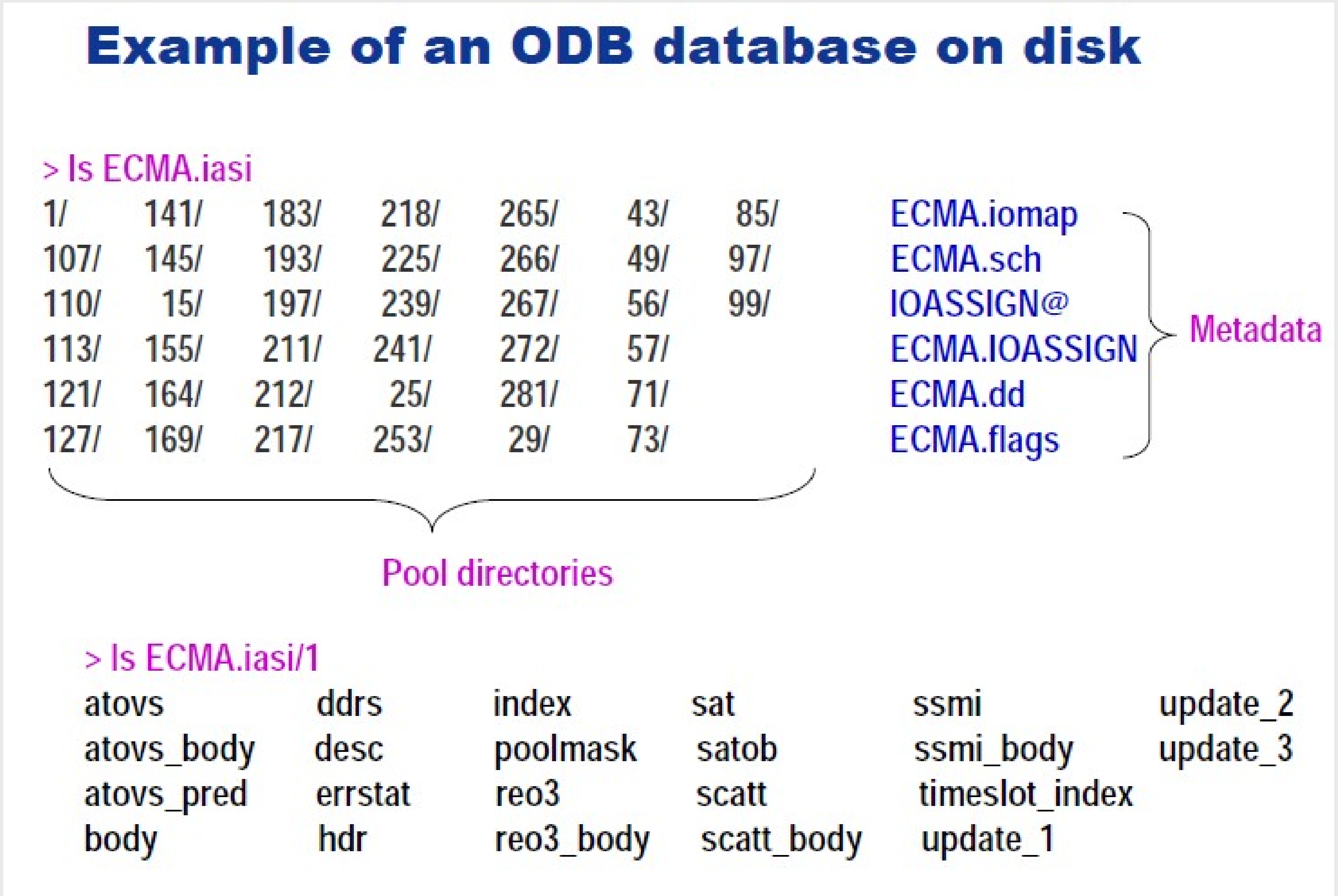
🟨 I/O tasks

- To improve performance, only a subset of pools is selected to perform I/O (read/write ODB on disk). Similar tables are then concatenated together.
- The number of I/O pools is fully configurable

ODB and its usage at ECMWF Slide 15

ECMWF

Reference: Anne Fouilloux (ECMWF)



Example of an ODB database on disk

```
> ls ECMA.iasi
1/     141/   183/   218/   265/   43/   85/     ECMA.iomap
107/   145/   193/   225/   266/   49/   97/     ECMA.sch
110/    15/   197/   239/   267/   56/   99/     IOASSIGN@
113/   155/   211/   241/   272/   57/          ECMA.IOASSIGN
121/   164/   212/    25/   281/   71/          ECMA.dd
127/   169/   217/   253/    29/   73/          ECMA.flags
```

Metadata

Pool directories

```
> ls ECMA.iasi/1
atovs        ddrs      index       sat          ssmi            update_2
atovs_body   desc      poolmask    satob        ssmi_body       update_3
atovs_pred   errstat   reo3        scatt        timeslot_index
body         hdr       reo3_body   scatt_body   update_1
```

ECMA – Extended Central Memory Array; CCMA – Compressed Central Memory Array

Reference: Anne Fouilloux (ECMWF)

ODB queries

In obsmon a query for satellite data looks like this:

```
SET $obstype=-1;
SET $varno=-1;
SET $sensor=-1;
SET $press=-1;
SET $statid=-1;

CREATE VIEW obsmon_sat
SELECT
obstype@hdr,codetype@hdr,satellite_identifier,varno,lat@hdr,lon@hdr,vertco_type@body,
vertco_reference_1@body,sensor@hdr,date,time@hdr,report_status.active@hdr,report_status.
blacklisted@hdr,report_status.passive@hdr,report_status.rejected@hdr,datum_status.active@body,
datum_status.blacklisted@body,datum_status.passive@body,datum_status.rejected@body,
datum_anflag.final,an_depar,fg_depar,obsvalue,final_obs_error@errstat,biascorr_fg,lsm@modsurf
FROM hdr,body,modsurf,errstat,sat WHERE
( obstype@hdr = $obstype ) AND
( varno@body = $varno ) AND
( sensor@hdr = $sensor ) AND
( vertco_reference_1@body = $press ) AND
( satellite_identifier = $statid ) AND
( an_depar IS NOT NULL )
```

ODB queries

In obsmon a query for conventional data looks like this:

```
SET $obstype=-1;
SET $varno=-1;
SET $press1=-1;
SET $press2=-1;
SET subtype1=-1;
SET subtype2=-1;

CREATE VIEW obsmon_conv
SELECT
obstype@hdr,codetype@hdr,statid,varno,lat@hdr,lon@hdr,vertco_type@body,vertco_reference_1@b
ody,sensor@hdr,date,time@hdr,report_status.active@hdr,report_status.blacklisted@hdr,report_status
.passive@hdr,report_status.rejected@hdr,datum_status.active@body,datum_status.blacklisted@body
,datum_status.passive@body,datum_status.rejected@body,datum_anflag.final,an_depar,fg_depar,obs
value,final_obs_error@errstat,biascorr_fg,lsm@modsurf
FROM hdr,body,modsurf,errstat WHERE
( obstype@hdr = $obstype ) AND
( varno@body = $varno ) AND
( codetype@hdr >= $subtype1 ) AND
( codetype@hdr <= $subtype2 ) AND
( vertco_reference_1@body > $press1 ) AND
( vertco_reference_1@body <= $press2 ) AND
( an_depar IS NOT NULL )
```

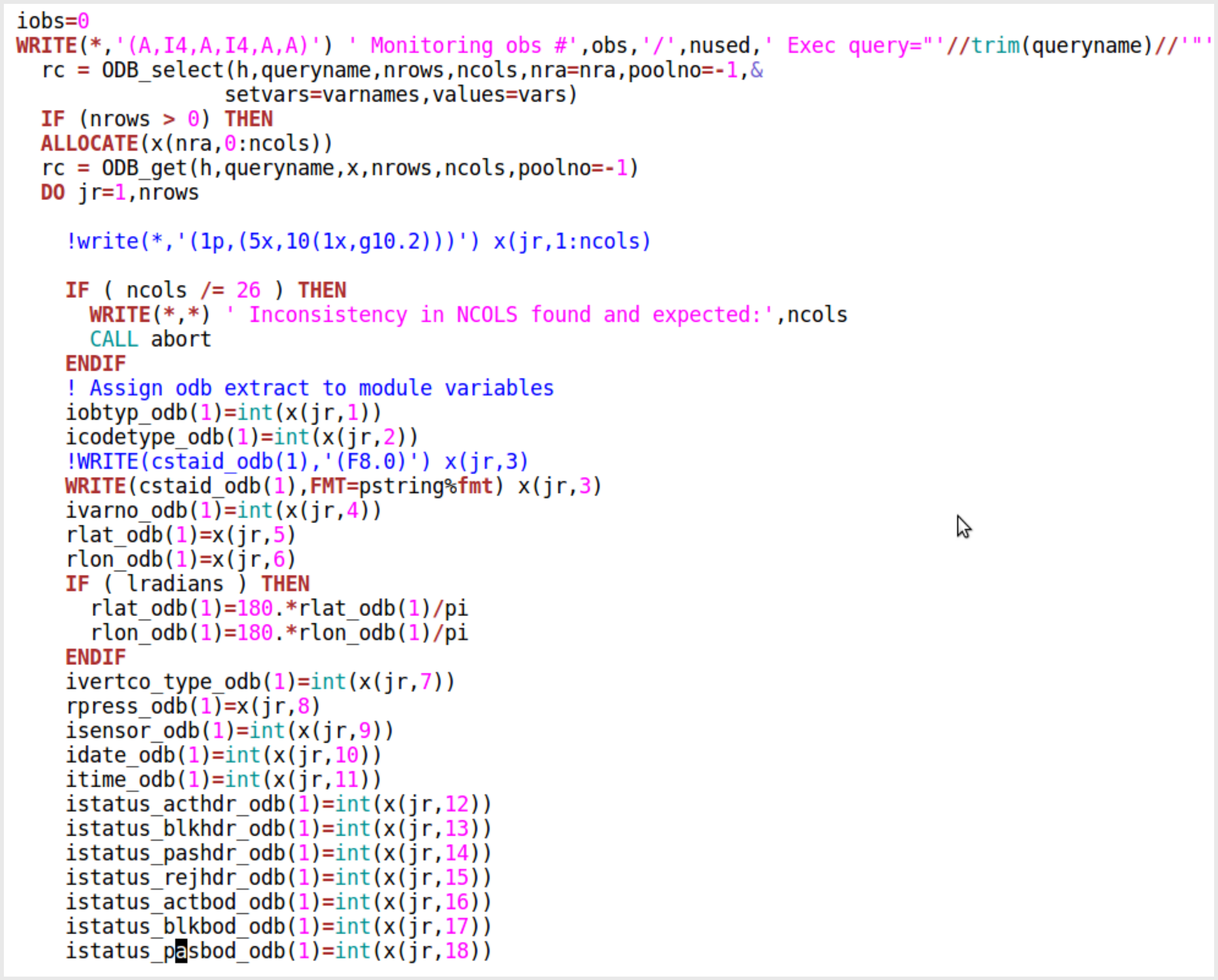
ODB queries

In obsmon a query for conventional data looks like this:

```
SET $obstype=-1;
SET $varno=-1;
SET $press1=-1;
SET $press2=-1;
SET subtype1=-1;
SET subtype2=-1;

CREATE VIEW obsmon_conv2
SELECT
obstype@hdr,codetype@hdr,statid,varno,lat@hdr,lon@hdr,vertco_type@body,vertco_reference_2@b
ody,sensor@hdr,date,time@hdr,report_status.active@hdr,report_status.blacklisted@hdr,report_status
.passive@hdr,report_status.rejected@hdr,datum_status.active@body,datum_status.blacklisted@body
,datum_status.passive@body,datum_status.rejected@body,datum_anflag.final,an_depar,fg_depar,obs
value,final_obs_error@errstat,biascorr_fg,lsm@modsurf
FROM hdr,body,modsurf,errstat WHERE
( obstype@hdr = $obstype ) AND
( varno@body = $varno ) AND
( codetype@hdr >= $subtype1 ) AND
( codetype@hdr <= $subtype2 ) AND
( vertco_reference_2@body > $press1 ) AND
( vertco_reference_2@body <= $press2 ) AND
( an_depar IS NOT NULL )
```

Reading ODB in Fortran

queryname=”obsmon_sat” or “obsmon_conv”

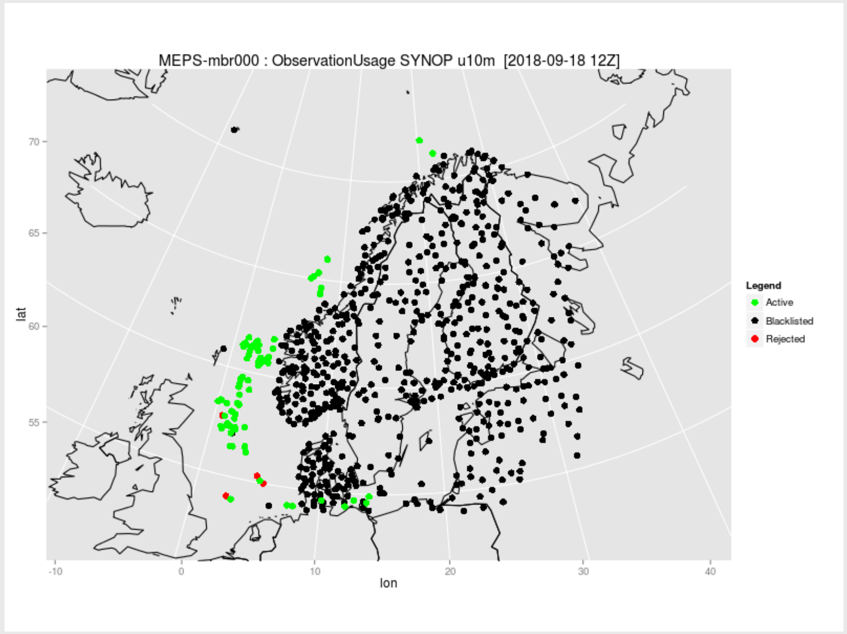


```
iobs=0
WRITE(*,'(A,I4,A,I4,A,A)') ' Monitoring obs #',obs,'/',nused,' Exec query="'//trim(queryname)//'"'
  rc = ODB_select(h,queryname,nrows,ncols,nra=nra,poolno=-1,&
                 setvars=varnames,values=vars)
  IF (nrows > 0) THEN
  ALLOCATE(x(nra,0:ncols))
  rc = ODB_get(h,queryname,x,nrows,ncols,poolno=-1)
  DO jr=1,nrows

    !write(*,'(1p,(5x,10(1x,g10.2)))') x(jr,1:ncols)

    IF ( ncols /= 26 ) THEN
      WRITE(*,*) ' Inconsistency in NCOLS found and expected:',ncols
      CALL abort
    ENDIF
    ! Assign odb extract to module variables
    iobtyp_odb(1)=int(x(jr,1))
    icodetype_odb(1)=int(x(jr,2))
    !WRITE(cstaid_odb(1),'(F8.0)') x(jr,3)
    WRITE(cstaid_odb(1),FMT=pstring%fmt) x(jr,3)
    ivarno_odb(1)=int(x(jr,4))
    rlat_odb(1)=x(jr,5)
    rlon_odb(1)=x(jr,6)
    IF ( lradians ) THEN
      rlat_odb(1)=180.*rlat_odb(1)/pi
      rlon_odb(1)=180.*rlon_odb(1)/pi
    ENDIF
    ivertco_type_odb(1)=int(x(jr,7))
    rpress_odb(1)=x(jr,8)
    isensor_odb(1)=int(x(jr,9))
    idate_odb(1)=int(x(jr,10))
    itime_odb(1)=int(x(jr,11))
    istatus_acthdr_odb(1)=int(x(jr,12))
    istatus_blkhdr_odb(1)=int(x(jr,13))
    istatus_pashdr_odb(1)=int(x(jr,14))
    istatus_rejhdr_odb(1)=int(x(jr,15))
    istatus_actbod_odb(1)=int(x(jr,16))
    istatus_blkbod_odb(1)=int(x(jr,17))
    istatus_pasbod_odb(1)=int(x(jr,18))
```

Reading ODB using tools – odbsql

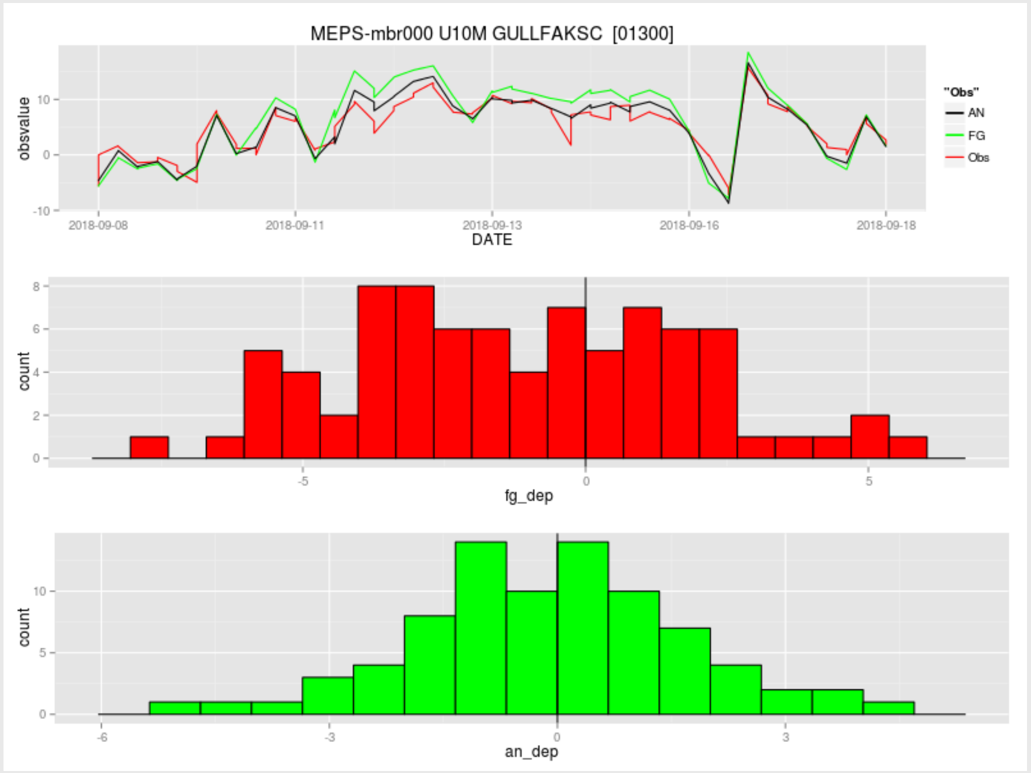
```
cd odb_ccma/CCMA
dcagen -F -n -N 1
odbsql -q "SELECT obstype, statid, varno, vertco_reference_1, sensor, scanpos, an_depar,
fg_depar, obsvalue, biascorr_fg, lsm  FROM modsurf, hdr, desc, body, radiance
WHERE (obstype == 7) AND (sensor /= 16) AND (lsm >= 0.1) ;"  -o output.dat
```

Visualisation of the monitoring through obsmon



Examples from

https://shiny.hirlam.org/obsmon2/


MEPS-mbr000 U10M GULLFAKSC [01300]
obsvalue
DATE
"Obs"
AN
FG
Obs
count
fg_dep
count
an_dep