ODB Management in ALADIN

Kertész Sándor

(Hungarian Meteorological Service)

1st ALADIN Maintenance Workshop

25-28 November 2002, Budapest

Outline

- Basic ODB structures
- Data retrieval
- OBB in ARPEGE/IFS
- ODB related programs
- Working with ODB

What is ODB?

- ***ODB = Observational DataBase***
- Relational database software developed at ECMWF (Sami Saarinen)
- Fast data retrieval by data query language ODB/SQL
- Allows flexible data layout definition

About ODB

- Used instead of CMA files in ARPEGE/IFS
- Written in FORTRAN90 and C. The low level API is quite difficult (hash tables etc.).
- Problems:
  - only a few documents
  - no technical documentation (black box)
  - hard to port

Basic data structures

- Data layout consists of uniquely named **TABLE**s
- Tables are made up of uniquely named **COLUMN**s
- Notation: *column_name@table_name*
- Up to now 23 tables are defined in ODB. E.g.:
  - hdr : the header of a SYNOP, TEMP etc. report
  - body: all information of one observed value ...

Definition of table hdr

```
CREATE TABLE hdr AS (
    seqno int,              // observation sequence
    obstype pk5int,         // observation type
    …
    date YYYYMMDD, // obs. date
    time HHMMSS,    // obs. time
    …
    ident pk5int,           // station id
    lat real8,              // latitude
    lon real8,              // longitude
    …
    );
```

Predefined variables

$SYNOP	1	SYNOP, SYNOP_SHIP, SYNOR
$AIREP	2	AIREP, AMDAR, ACAR, CODAR, COLBA
$NSATOB	3	SATOB
$NDRIBU	4	DRIBU, DRIFTER, BUOY, BATHY, TESAC
$NTEMP	5	TEMP, TEMP-SHIP, TEMP_DROP, ROCOB

$u	3	u (composante zonale du vent) m s-1
$v	4	v (composante méridienne du vent) m s-1
$z	1	géopotentiel J.kg-1
$dz	57	épaisseur de couche (SATEM) J.kg-1
$rh	29	humidité relative de niveau de 0 ŕ 1
$pwc	9	contenu en eau précipitable kg.m2
$rh2m	58	humidité relative ŕ 2 mčtres de 0 ŕ 1

What data types?

- integer, real (packed or unpacked)
- string
- YYYYMMDD, HHMMSS (storage of date)
- compound types for bitfields
- @LINK type variables (relations between tables)
- arrays for any data type

Bitfield data types

- Maximum 32 one-bit members per type
- One member can be at most 16 bit length

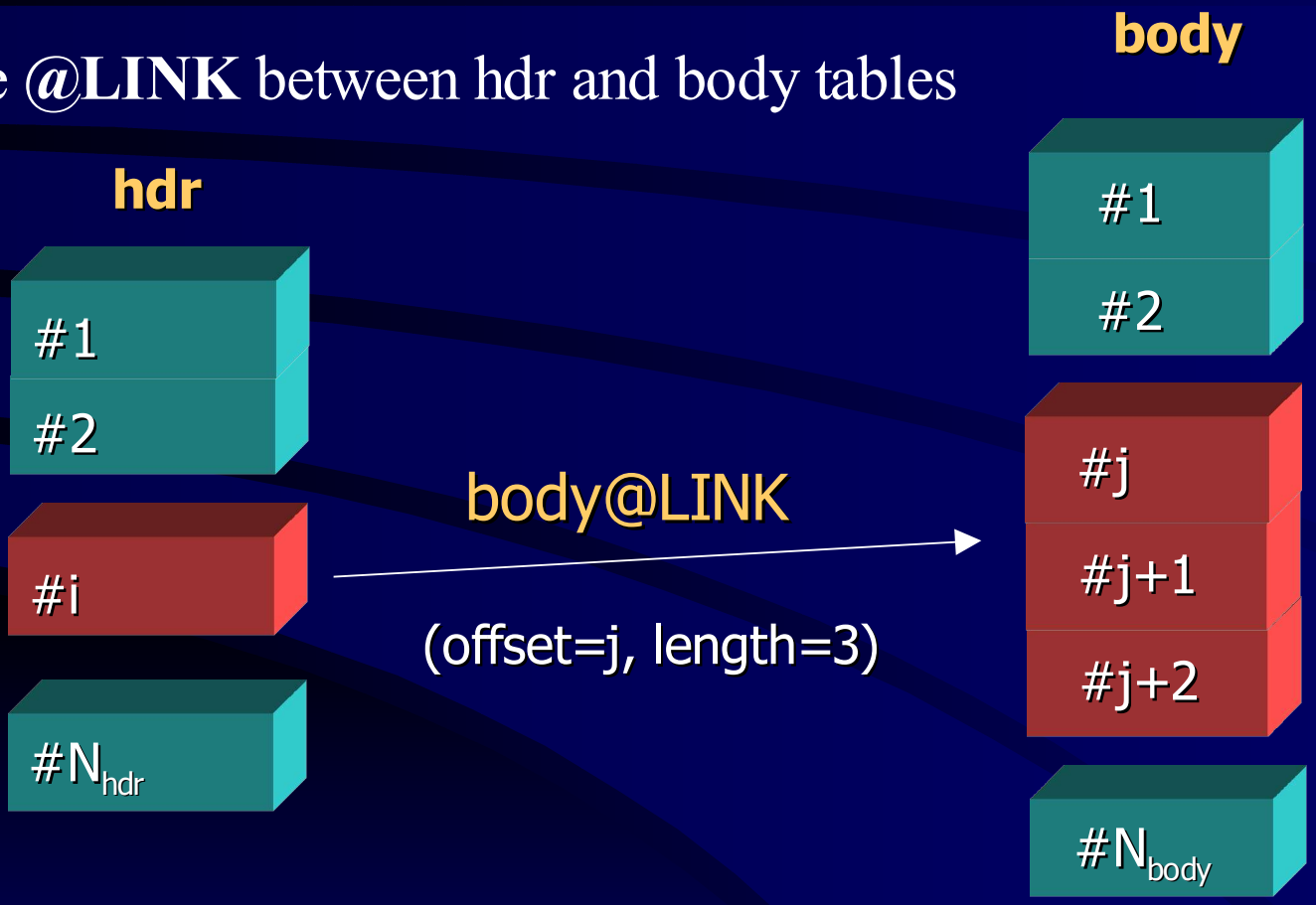
```
CREATE TYPE status_t AS (
    active bit1,
    passive bit1,
    rejected bit1,
    blacklisted bit1,
    … );
```

@LINK data types

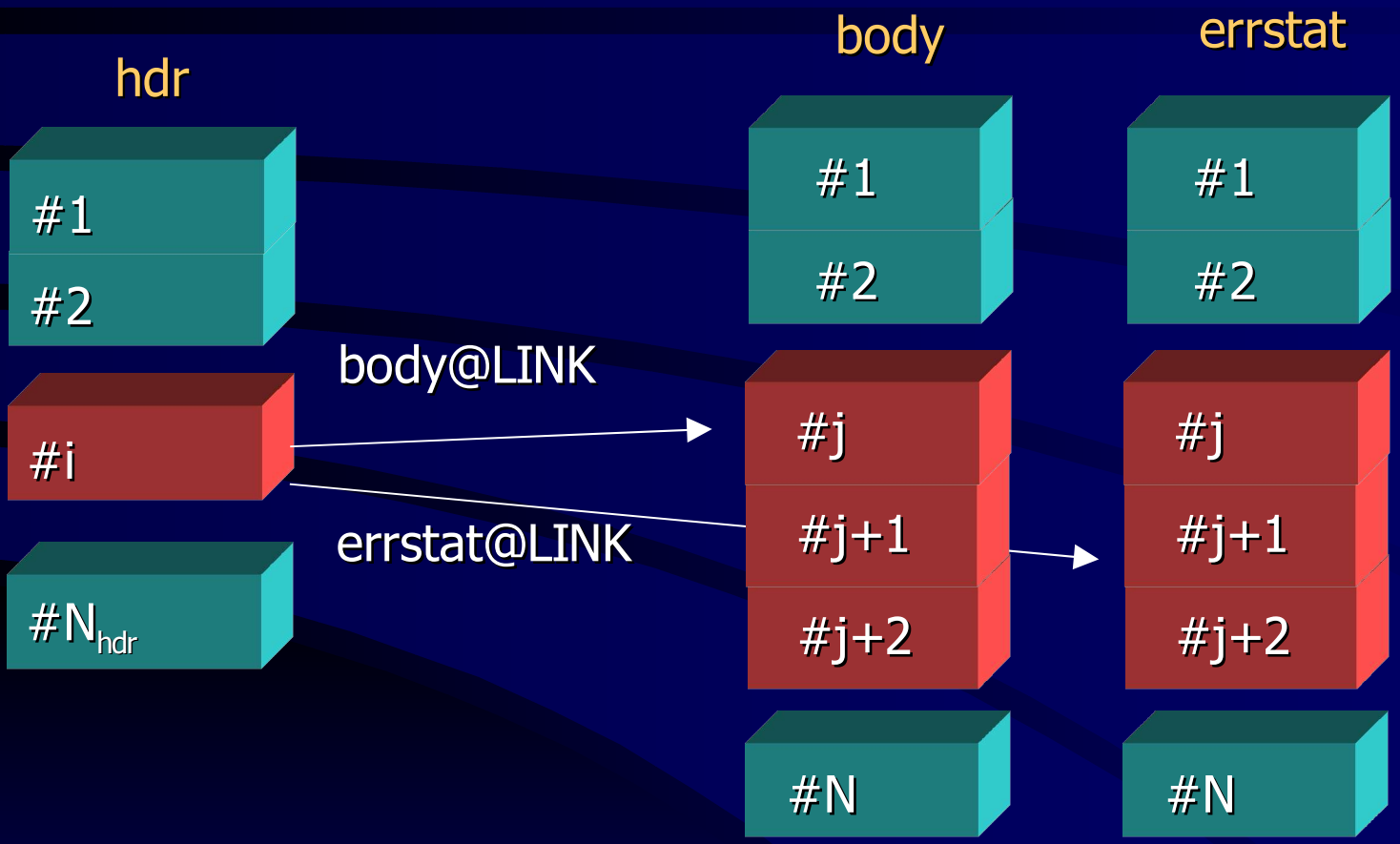
- Let's imagine a "thin" database with only N SYNOP reports with u, v, T observations. This database contains:
  - N hdr table for report headers
  - 3*N body tables for data (T, u, v)
- **body@LINK** in hdr will identify the body tables that belong to the given hdr table

@LINK data types

- The **@LINK** between hdr and body tables



Aligned tables



Database types

- A set of table types forms a database type
- The most important ones:
  - **ECMA**: all the tables (22) needed for screening ↔ Extended CMA files
  - **ECMASCR**: another name for ECMA. It is used in special ODB related programs.
  - **CCMA**: all the tables (14) needed for minimisation ↔ Compressed CMA files

Pools

- Data partitioning for multiprocessor-runs is realized via the so called pools
- A pool contains a certain amount of data in the database
- Partitioning can be done with respect to latitude-bands and/or timeslots
- **Number of pools >= Number of processors**

What does ODB look like?

- For a given database type data for different analysis dates are stored in separate directories
- Such a directory can be anywhere in the filesystem and contains:
  - *dbtype*.dd: (ECMA.dd etc.) miscellaneous default values
  - *dbtype*.sch: (ECMA.sch etc.) scheme of the database
  - one directory for each pool that contains one file for each table of the given database type

Data retrieval

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

```
CREATE VIEW test1 AS
SELECT
    obstype, ident,
    lat, lon,
    varno
FROM hdr, body
```

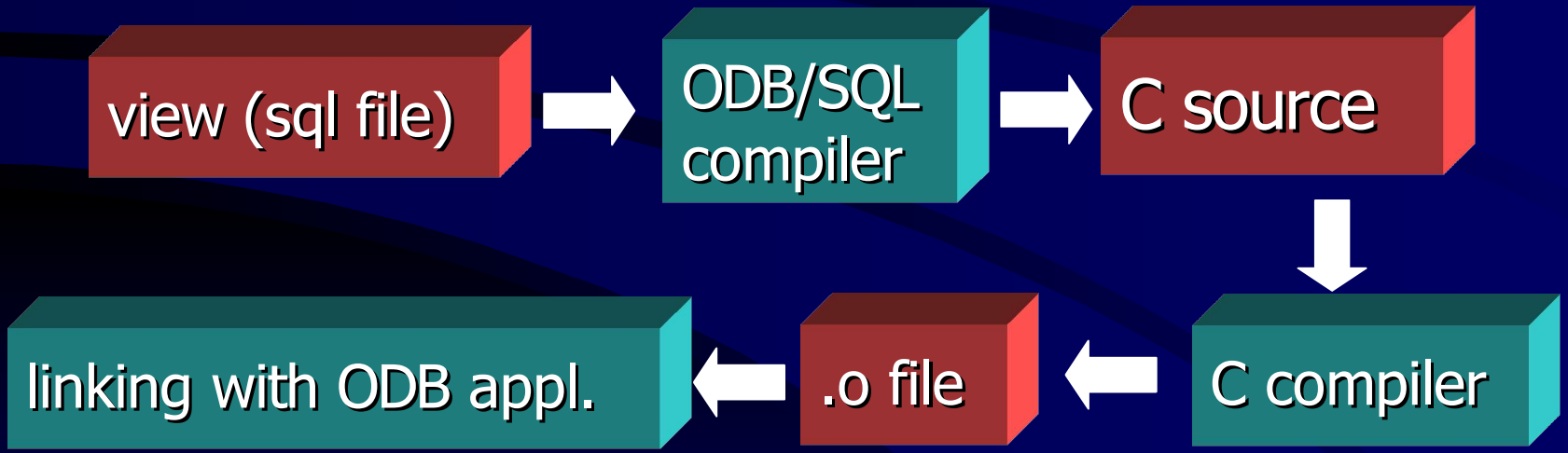
Data retrieval

- Additional query options:

```
CREATE VIEW test2 AS
SELECT obstype, ident,
            lat, lon, varno
FROM hdr, body
WHERE obstype = 1
    AND status.active@body
ORDERBY ident DESC
```

Data retrieval

- Views are placed into .sql ascii files
- These files cannot be used directly by ODB. A special compilation is needed:



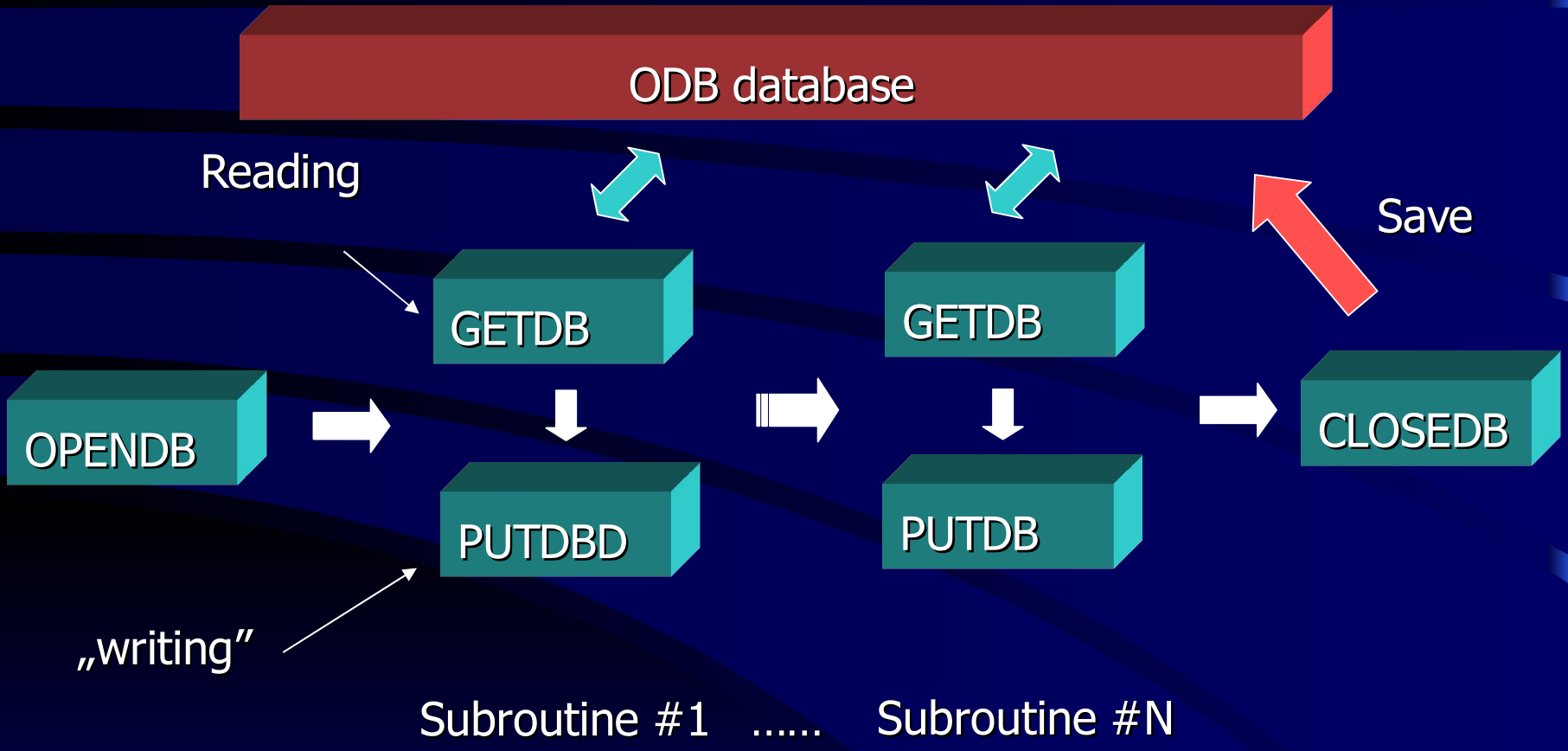
ODB in ARPEGE/IFS

- Implemented in CY22R3. AL15 doesn't use CMA files.
- ROBSAR observation vector is no longer used
- Intermediate layer of F90 subroutines that communicates through module **YOMDB**
- Performance within +/-1% that of ROBSAR

ODB in ARPEGE/IFS

- For the model level interfce we have to know only four subroutines:_
  - **OPENDB**
  - **GETDB**
  - **PUTDB**
  - **CLOSEDB**

ODB in ARPEGE/IFS



GETDB-PUTDB

- GETDB retrieval is based on **views** defined for the given subroutine: only the neccessary data is read.
- Data is read into dynamically allocated matrices
- Subroutine then works with the retrieved data
- In the end PUTDB deallocates the matrices
- ODB data in the model **only available** between GETDB-PUTDB calls!

Retrieval with GETDB

- Target matrices are defined in module YOMDB (MOB…=integer4, ROB…=real8):
  - MOBHDR, ROBHDR: report header data
  - MOBODY, ROBODY: report body data
  - MOBSU, ROBSU: setup related data
  - ROBDDR: DDR related data
  - SATHDR, SATBODY, etc.: miscellaneous satellite specific data

Retrieval with GETDB

- Each ODB column name has an integer counterpart in YOMDB. E.g.:
  - lat@hdr → MDBLAT
- If the target matrix is ROBHDR then

ROBHDR(i,MDBLAT)

will refer to the lat@hdr value of the i-th report in the extracted data

- Keep in mind: MDBLAT is set by GETDB!

Contexts

- One of the arguments of GETDB is the so-called **context**
- Each observation related subroutine has its own context that is defined in ctxinitdb.F90
- A context specifies:
  - a group of views
  - target matrices for the views
  - updateable columns of the views

MPI Problems

- Initialize MPI
- Problems in screening: MPI is initialized 2 times
- Buffered communication parameters are set in NAMPAR for screening, but no settings for ODB → „Out of buffer" space problem
- ODB hack

ODB related programs

- Execution is controlled by a set of env. shell variables. The most important ones:
  - ODB_CMA: specifies the database type
  - ODB_SRCPATH_*dbtype*: specifies the location of the .dd and .sch files of
  - ODB_DATAPATH_*dbtype*: specifies the location of the pools
  - IOASSIGN: specifies the ioassign file that describes the directory structure of the database

Program: ODBTOOLS

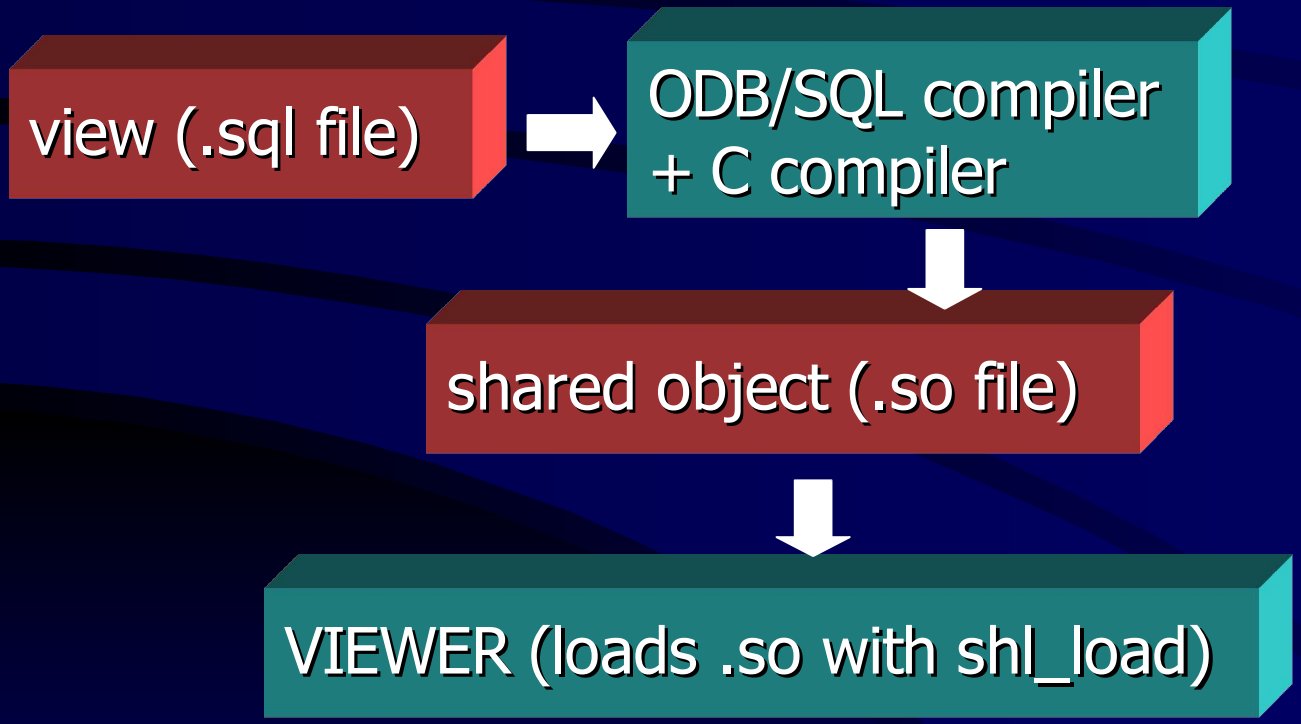
- The main ODB handling program provided by the ODB package (TRANSFODB at MF)
- It has three modes that are identified by the actual program name:
  - TO_ODB: CMA file → ODB conversion
  - TO_ ECMA: ODB → CMA file conversion
  - SHUFFLE: ODB → ODB conversion and data partitioning for parallel runs

Program: ODBTOOLS

- Tasks of SHUFFLE:
  - changes the number of the pools
  - defines timeslots and time window
  - selects only the marked data
  - ECMASCR → ECMA: data partitioning and selection
  - ECMA → CCMA: data selection after screening, creates input for minim.
  - CCMA → ECMA: new information after minim. is put into the "parent" ECMA database

Program: VIEWER

- "Dynamic" retrieval based on user defined views

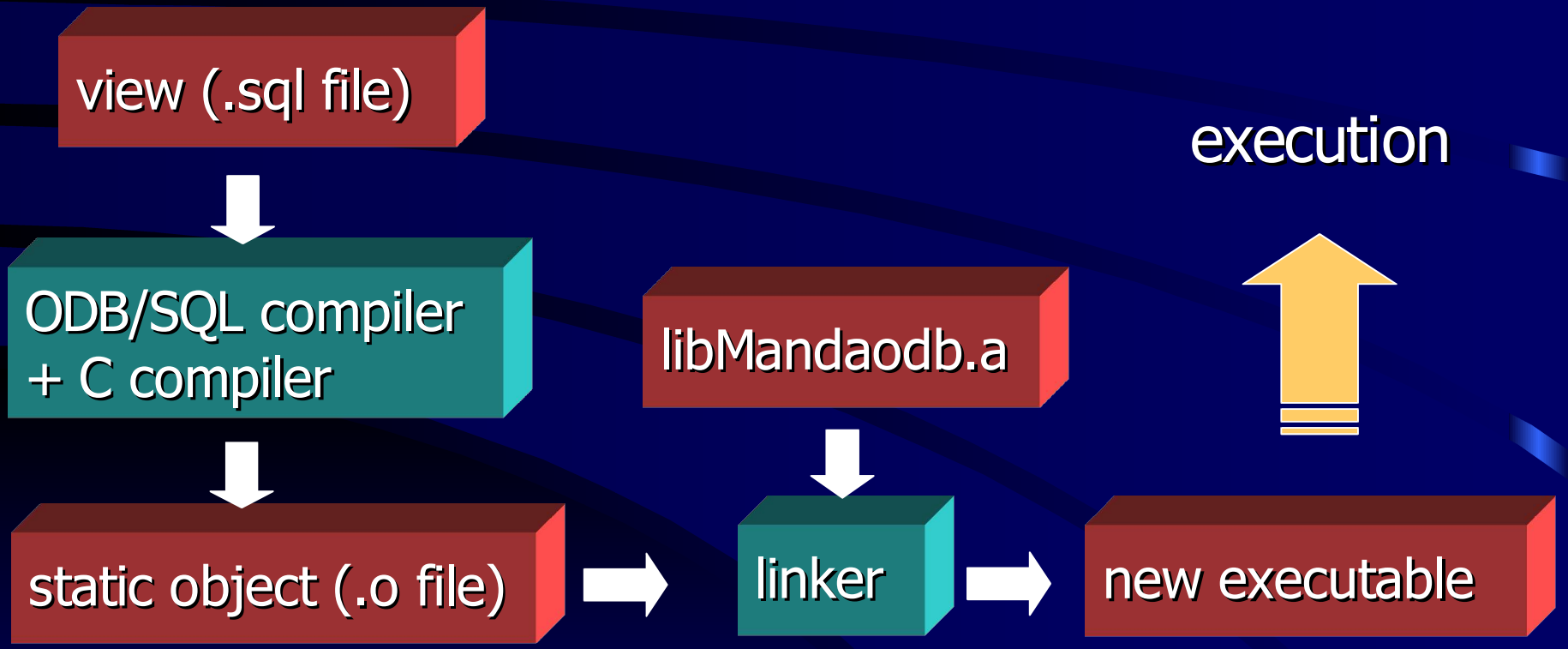


Program: MANDAODB

- Data is retrieved into an ascii file
- Retrieval is based on predefined (at present there are 3) and user defined views
- Continuously evolving (Dominique Puech)
- In the future it will integrate many task of MANDALAY (dumping obs. related info in different formats)

Program: MANDAODB

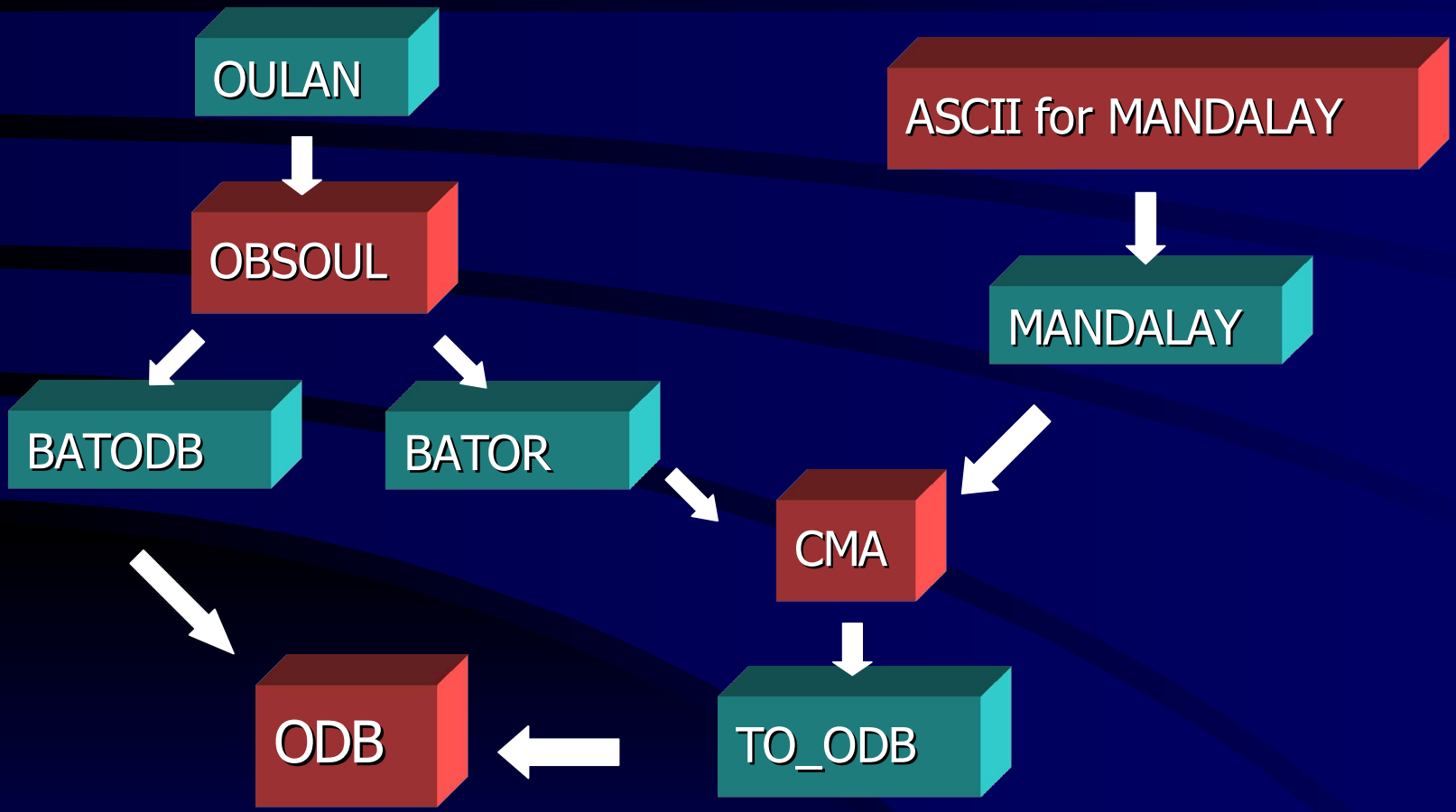
- "Static" retrieval based on user defined views



Creating ODB

- Getting ODB from Meteo France: global ECMASCR ODB available operatively since 1st September 2001
- Conversion from a CMA file
- Conversion from an OBSOUL file

Creating ODB



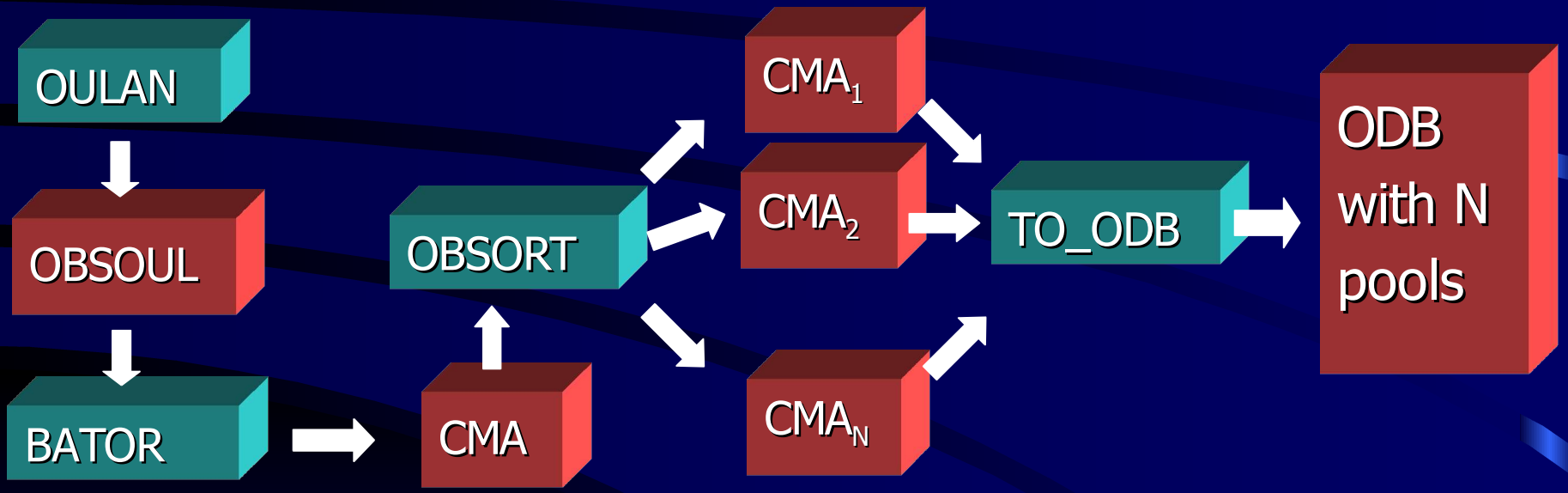
Creating ODB

- Problems with MANDALAY:
  - the created ODB cannot converted back to CMA with TO_ECMA
  - the created ODB cannot used by screening because some DDR entries are not set → OBSORT must be put between MANDALAY and TO_ODB
- With BATOR or BATODB we haven't got these problems, but BATODB can be run only on 1 proc. (at HMS)!

Creating ODB at HMS

- One 3dvar script that contains all the execution steps (from obs. preparation to analysis)
- Due to technical problems all programs have to be run on the same number of PE
- Problems:
  - BATOR must be used instead of BATODB
  - on N procs. TO_ODB needs at least N CMA files that cannot be produced with BATOR

Creating ODB at HMS

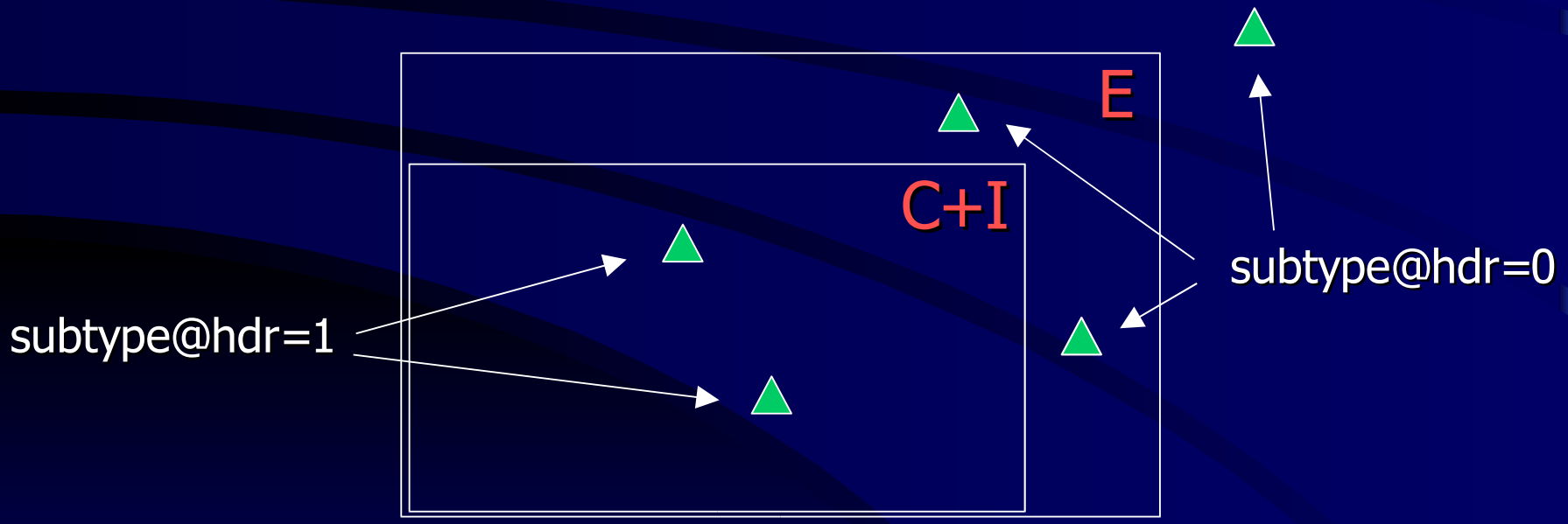


3DVAR with ODB

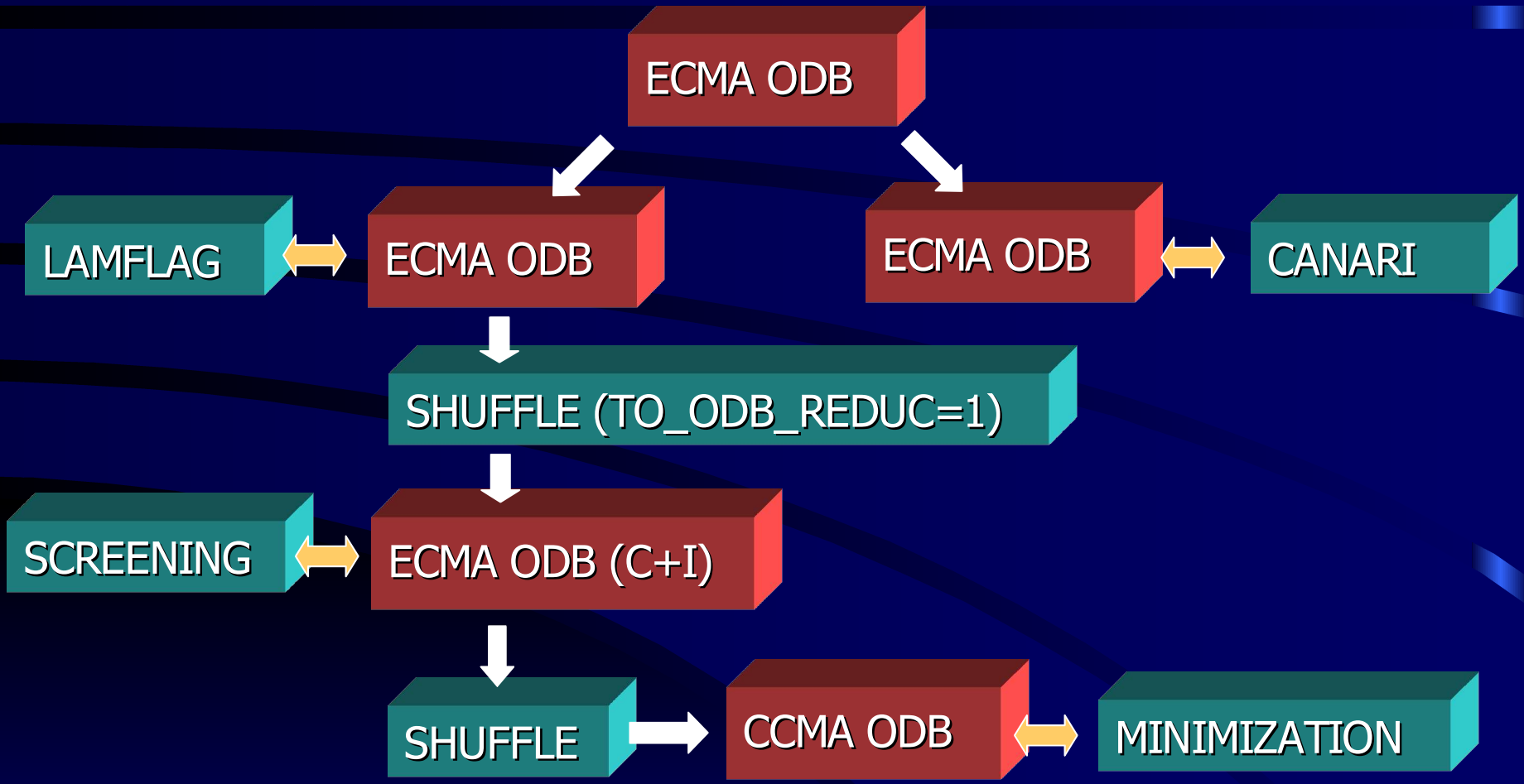
- Let's suppose we have an ECMA ODB with the required number of pools. we want execute a 3dvar cycle: screening, minimization and canari
- Problem: screening crashes if there are observations outside the C+I zone in the ODB
- The simplest solution: create a new input ODB for screening with the right observations only

3DVAR with ODB

- Program **LAMFLAG** marks the right observations, then SHUFFLE with TO_ODB_REDUC=1 creates a new ODB containing only the right obs.



3DVAR with ODB



What is next?

- Biggest problem is the lack of documentation, but this will be solved soon
- Little endian version
- More experts are needed
- Try to forget CMA files and MANDALAY
- Development of MANDAODB
- Visualization: ODB interface for metview