OPERATIONAL DATABASES AT MÉTÉO-FRANCE

Michel Pottier
Météo-France
Toulouse, France

Summary: I present here, the new operational databases of Météo-France. It has been implemented using Oracle RDBMS and an adaptation of the NEONS software.

1 CHOICE OF ORACLE AND NEONS

At the end of 1992 Météo-France took the decision to port its operational system from Control Data / NOS to UNIX. We also took the decision to use RDBMS to manage the data. In May 93, we took the decision to use NEONS and Oracle. In July the port of the original version of NEONS from Empress to Oracle was made. In August we began to adapt NEONS to our needs, that means to the needs of an operational system. In February 94, the LLT datatype was fully operational. This datatype has never been modified since this date.

In April 95 our new operational system Diapason was operational with a complete new version of NEONS including :

- Grib, lltbufr, forecast, flag datatypes
- A set of programs to manage primary tables
- An archive on optical jukebox
- A set of tools to replicate data between the different machines, to backup the data, to rebuild the database, ...

- The choice of NEONS has been made because:
  - We thought that we will be able to go faster.
  - It is a good solution to manage a big volume of data.
  - And it allows a transparent access to the archive.
- We chose to use a RDBMS because:
  - RDBMS are, now, very efficient, and available on many platforms.
  - SQL is a standard and enables very sophisticated requests.
  - RDBMS include mechanisms to manage integrity and security of the data.
  - They are, now, able to manage all kind of data (Blob).

It is important to note that the port of NEONS from Empress to Oracle was completed in 3 weeks, and 6 months later the LLT datatype (observational reports) was operational.

2 MÉTÉO-FRANCE OPERATIONAL DATABASES

There are two operational machines, and on each of them there is one database. At each time, the status of one is operational and the status of the other is backup. At each moment we can switch the status of the two machines. Each database has the same NEONS structure which includes 5 datatypes and for each of them the three realms associative, descriptive and primary. The available datatypes are:

- Point forecasts (surface and upper air) produced by models or forecasters.
- Quality/Control flags put on the observations by models or forecasters.
- Observations (30 different sequence types have been defined)
- Numerical Models Outputs.
- Image (will be made in the near future).

A data model is also available to manage all the world stations. This model has been designed according to the Volume A of the WMO.

The volume of one operational database is 45Gb. A third database, the archive database has the same design and Oracle manages a volume of 200Gb of optical disks. A transparent access to the archive is possible due to the conceptual data model of NEONS. For each of these databases :

- A management of confidentiality is available according to the product, the producer and the hour of the production.
- There is a management of availability of "resources" for the monitor.
- The storage of data is made in meteorological format (Grib, Bufr), or according to the standard relational model (forecasts, Q/C flags), or in mixing the two methods (observations).
- For each datatype there is only one routine to access to the data independently of the structure of the primary table. These C routines have been wrote using dynamic SQL of Oracle, so they allow the user to write true SQL (''from'', ''where'' and ''order by'' clauses).

- A set of tools has been developed:
  - To ingest data in real time. A file format has been defined between the preprocessing and the routines which insert data, so it is very easy to adapt this version of NEONS to any preprocessing.
  - To manage all the primary tables.
  - To supervise database and all the process.
  - To archive data and to be sure that the different associative tables are consistent with the archive dataset.
  - To backup, and restore the descriptive and associative tables.
  - To create a new database.
  - To replicate data between a lot of databases. That especially concerns the descriptive tables and the associative tables for archive datasets.
  - And Forms4 interfaces have been developed to manage the different data models. They allow, also, to display the contents of primary tables including the Bufr contents.

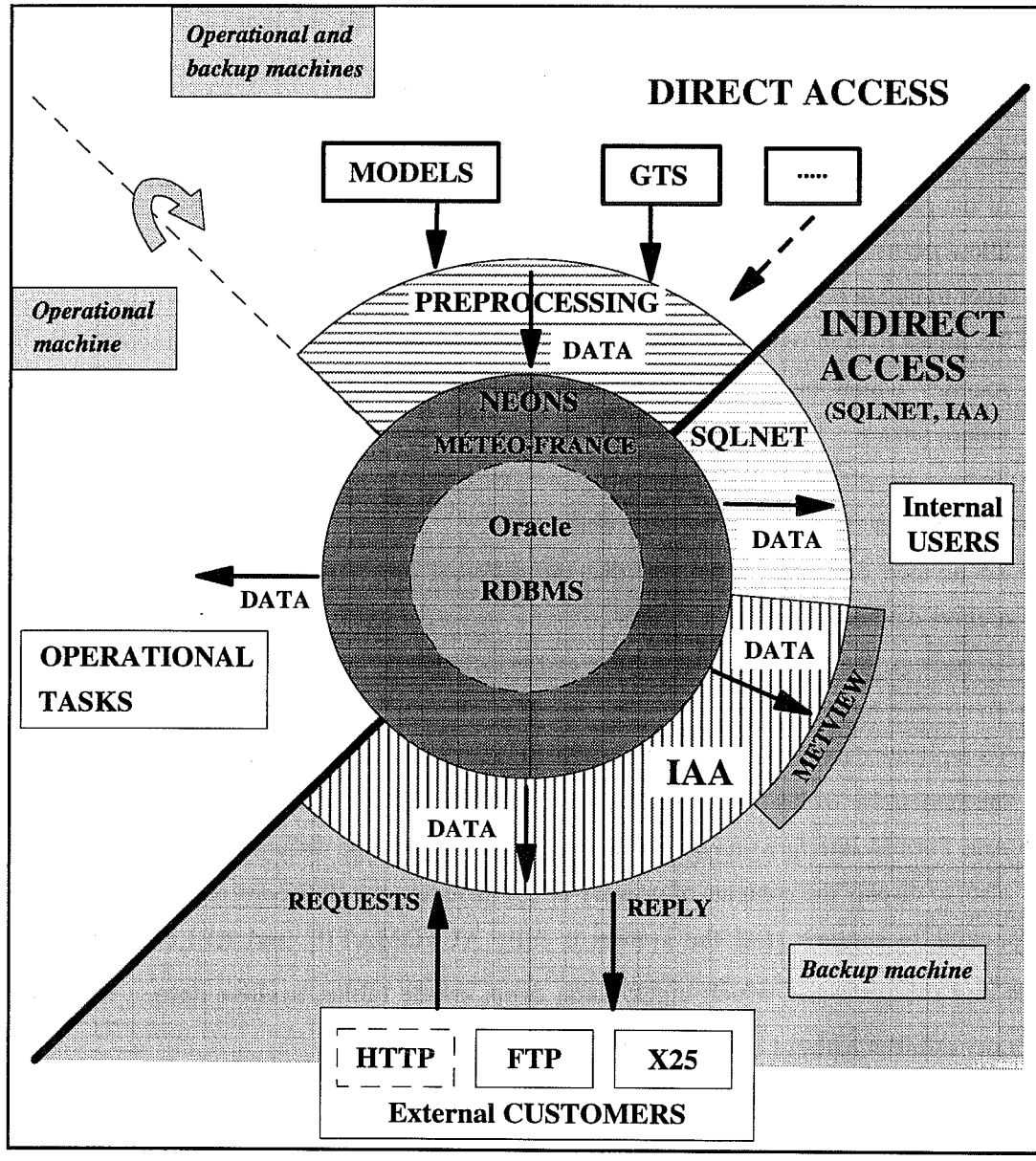
It is important to remark that the system is fully AUTOMATIC and RELIABLE. The Oracle processes and the processes which ingest data, drop, create tables, archive data, .. are able to run during many months without requiring any human intervention.

3 HOW THE OPERATIONAL DATABASES ARE ACCESSED

This graphic presents the different ways which are possible to access to the data. We can made a separation between the direct access, in white, and the indirect access in grey. In the ''direct access'' part we find mainly all the operational tasks:

- Operational tasks which ingest data arriving from the models or from the GTS or from other operational tasks. These tasks run on the two machines.
- Operational tasks which read data to make other elaborated products. They run on the machine which has the status ''*operational*''.

In the ''indirect access'' part we find the users and customers of Météo-France. The users or customers can access to the data using IAA (Random Access Interface) or SQLNET if they are developing future operational tasks. You will note that this ''indirect access'' access to the data which are on the ''*backup*'' machine and that internal users can use Metview to display all the meteorological data.



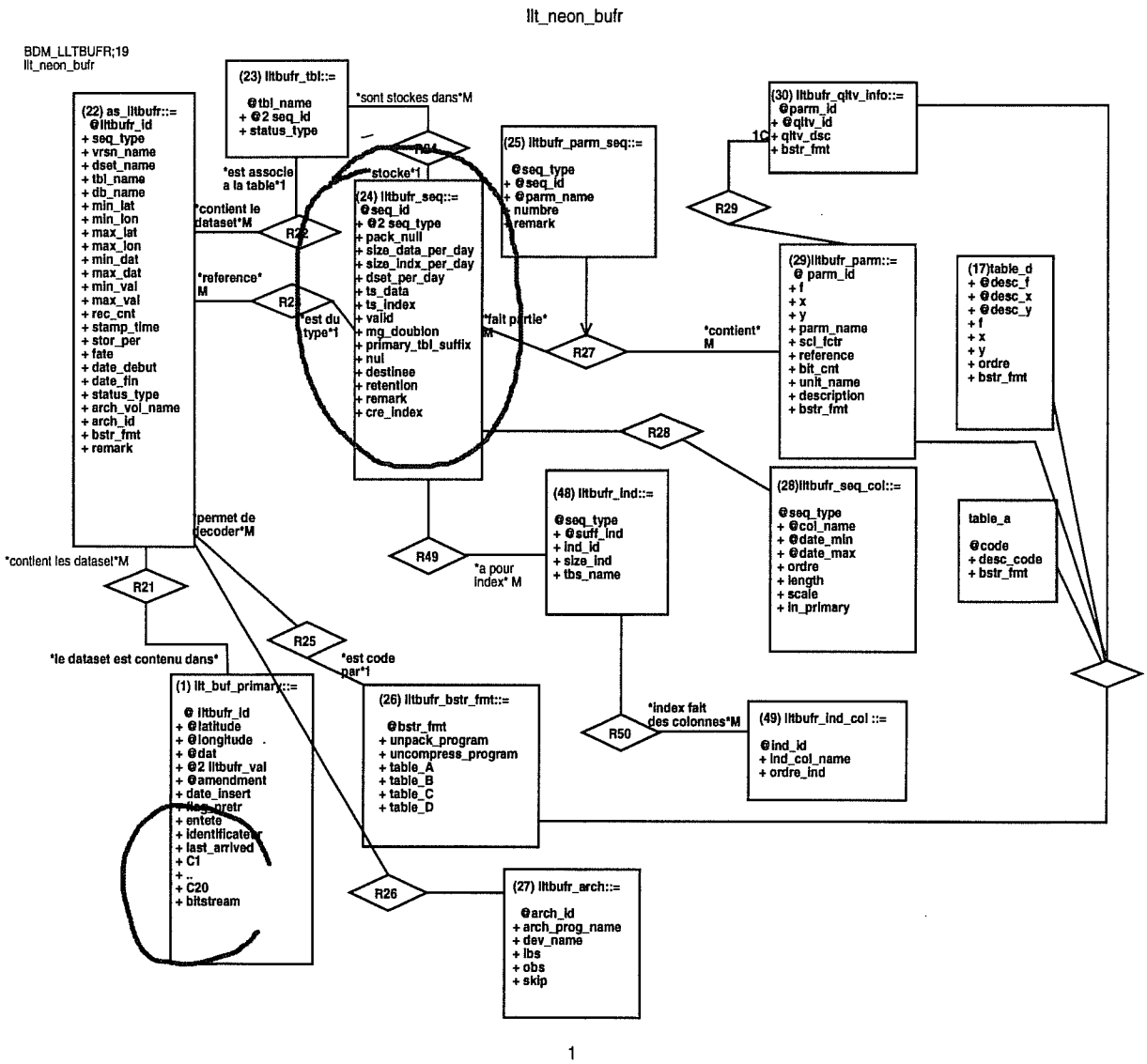
4 MÉTÉO-FRANCE VERSION OF NEONS : EXAMPLE; THE LLT DATATYPE

4.1 Description

Here you can see the information model for the lltbufr datatype. It is used to manage the observational reports. I present it as an example of our 5 datatypes. The name is lttbufr because in the column bitstream of the primary table, we find true Bufr.

With this model we manage more than 30 different sequence types. For each of these sequence types, there is one record in the table *lltbufr_seq*. In this record you find information :

- On the place where the table must be created : attributes **ts_data** and **ts_index.**



- On the size of the primary table for this sequence type : attributes **size_data_per_day** and **size_index_per_day**. The place for the table and index is reserved when the table is created.
- On the number of datasets we must create for this sequence type (attribute **dset_per_day**)
- On the fate of the dataset, for example delete of archive (attribute **destinee**)

- If the sequence_type is valid or not (attribute **valid**)
- If we must create other index than the primary key (attribute **cre_index**)
- If the bitstream of the primary table can be null or not (attribute **null**).

One other thing to be noted is that in the primary table you can find up to twenty five ''denormalized'' columns. A denormalized column is a column which will contain anything you want. In most of case, it is a parameter of the message which is in direct access. For each sequence type we can have up to 25 parameters in direct access, those being different for each sequence type. For this reason we have also a table lltbufr_seq_col which describes for each sequence type the set of denormalized columns. Because bitstream can be null, we can have primary tables without bitstream and with only denormalized columns. You can also notice in the associative table the attributes **db_name**, **fate** and **status_type**. They are used to manage the dataset and to distribute the data between different databases.

4.2 Performance

Using our C routines to extract data we obtain the following results:

- Read the Bufr of 29186 Synop (one day) : 29s.
- Read the Bufr of 3500 Synop (one hour) : 5s.
- Read 10 denormalized columns of 3500 Synop : 4s.
- Read 10 denormalized columns of 29186 Synop :17s.

5 THE ARCHIVE DATABASE

We have seen that the associative tables as_lltbufr, as_flag and as_previ have a set of attributes to manage archive.

db_name The name of an oracle database link to another database (here only one database: the archive).

fate The fate of an operational dataset: when the storage period is finished if fate='delete' dataset is destroyed, if fate='archive' dataset is inserted in the archive database.

status_type Give the status of the dataset . It can take different values:

- 'load' : the dataset is operational, available to load data
- 'delete' : the dataset must be destroyed
- 'load_histo' : it is an archive dataset which is available to load (oracle tablespace not full).
- 'optical' : it is an archive dataset which is no more available to load data. It is on optical jukebox.

We have the possibility to distribute data between different databases using the column **db_name**. (If you want to be able to access data from any database, all databases must have the same associative table). One of the parameters of the read routine allows user to choose between an access to the data of the local database on which he is connecting, or an access to all the data of all the databases which have been declared. This system allows access at the same time to different datasets anywhere they are.

6 CONCLUSION

Now we have an operational database build from NEONS-Oracle which runs without problem since many months (45Gb + 200Gb for archive). The LLT datatype includes more than 30 meteorological sequence types and to add a new one, it takes only a few minutes: We just have to insert one record in one table. We have two new datatypes (forecast and flag). For each of these datatypes we manage ''denormalized'' columns. For example, for LLT, each sequence type has its own set of denormalized columns. But for each datatype there is still, only one routine to insert or read the data. This routine allows user to write true SQL to access to the data which can be distributed between as many databases as we want.

We have chosen Oracle. There is no more doubt about the reliability and the efficiency of this RDBMS, and we think that this choice has permitted to save time, not only during the development run, but also during the operational run because management is easier. Lastly we think that the use of RDBMS allows to follow the evolution of the technology.