Simple Data Base Structures at ECMWF

J. K. Gibson, M. Dragosavac and B. Raoult

European Centre for Medium-Range Weather Forecasts (ECMWF)

Reading, U.K.

REPORTS DATA BASE

1. INTRODUCTION

The ECMWF requirements for the management of observational data are relatively simple. In particular:

- data are required for complete observations, or for sets of complete observations over a small time/area domain;
- a standard, self defining representation form should be used;
- the principle, time critical access to the data is that required for the support of data assimilation, and as such is well defined.

The data representation form chosen to meet the middle requirement is FM 94 BUFR; observations which have been pre-processed and converted to this form are called reports; the structure used to retain reports on-line is called a reports data base (RDB).

2. REPORTS DATA BASE DESIGN

2.1 Conceptual Model

A three level model (see fig. 2.1) is used to define the major aspects of the data base design; the following principles influence this choice:

- a "report", defined as a data structure representing either a single observation, or a set of observations over a small time/area domain, is the basic unit of access;
- reports may be grouped by type and/or sub-type, representing sets of similar reports, and providing a typical unit of collective access;
- within a given type, the set of reports for a given range of valid times is a typical unit of access required in support of ECMWF's data assimilation scheme;
- time and type have greater significance in terms of data access in support of ECMWF's forecasting system than area, because the area of interest is global.

Thus, conceptually the reports data bases may be considered to have data base, type and time levels.

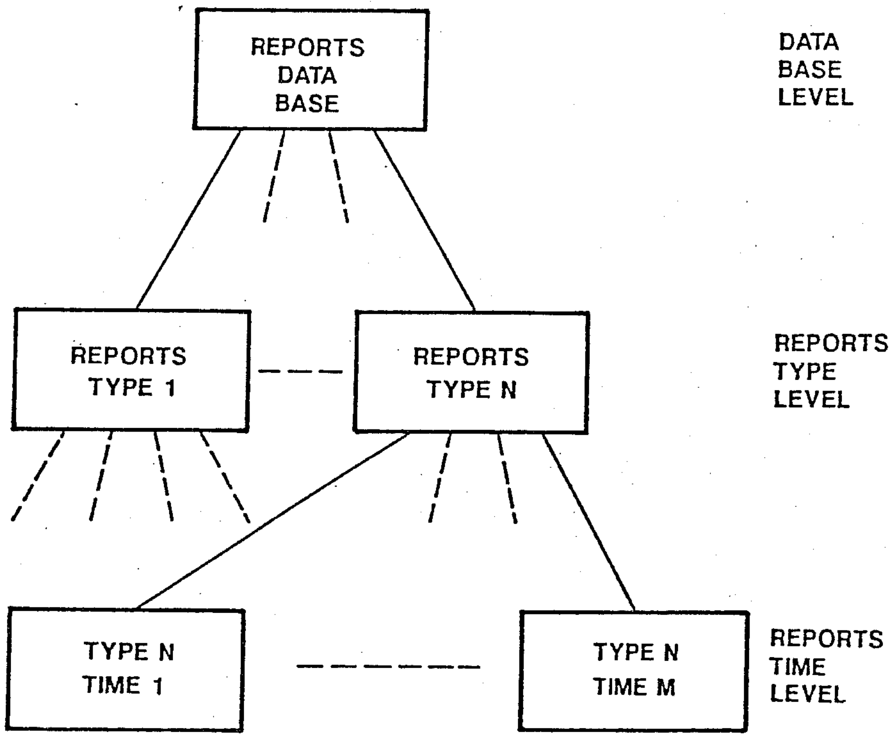

Fig. 2.1 Three level conceptual model

2.2 <u>General Conceptual Design</u>

At the DATA BASE level:

- one data base is required for each major implementation (operational system, test system, etc.)
- data base storage and access is supported by a single software package;
- write permission is restricted to selected, validated processes.

At the REPORTS TYPE level:

- one message type is mapped onto a distinct set of reports types;
- feedback information are classified by type to reflect both the type of data and the generating application;

At the REPORTS TIME level:

- each reports type is sub-divided into a set of reports times, comprised of reports of that type valid for a set range of times;
- the set range of times is chosen according to the reports type, and with relation to the requirements of the data assimilation system.

2.3 <u>Data Management Consequences</u>

The above conceptual design requires that the reports be organised in such a way that the smallest addressable object is a report, and that the smallest addressable collection of objects be a set of reports for a given time range of a given type within a given system.

The extraction of data for the data assimilation sub-system requires all reports for a given time range. The data assimilation may further require that different processes by presented with different report types. Since the most efficient extraction method is to read or copy complete files, it follows that the data collections at the reports time level should be individual files.

Such a solution is also effective with respect to the pre-processing. There exists one pre-processing task for each data type within the message data base; the concept of one message type being mapped onto a set of reports types thus results in each pre-processing task reading from, and writing to files which are distinct from those addressed by other pre-processing tasks.

It is both desirable and necessary that individual reports be identifiable and accessible. This is achieved by the generation of identification information called "keys". It is also desirable to retain certain processing, monitoring, or "housekeeping" information, such as the time of receipt of a report, the inclusion of various parts within a report, etc. Much of this information will need to be processed without reference to the data contained within the report itself, so should be separately accessible. At ECMWF such information is stored with the keys, which are then called "extended keys".

An additional consequence of the above is that the key information associated with any one file is likely to be small. Thus, provided the support software is run on a machine with a reasonable memory size considerable advantages would result from reading the keys into memory (or retaining them in memory), and performing searches on the memory resident copy.

Thus, the data management requirement of the physical implementation of a Reports Data Base is:

- reports, grouped in files according to type and time;
- key information, to identify each report;
- housekeeping information, to supplement the key information;
- access software.

3. IMPLEMENTATION

3.1 Introduction

RDB systems have been successfully implemented on two different platforms - on Digital Equipment Corporation machines running the VMS operating system, and within a Network File System under UNIX. Many of the basic principles followed are the same for both systems.

3.1 VMS Version

The physical implementation follows closely the conceptual model. The data base level is a master directory for the data base. The reports type level is a sub-directory for each type. The reports time level is a file for each range of valid times.

In order to make as much use as possible of the facilities offered by the operating system, files are indexed sequential in type, accessible by key (see fig. 3.1). Keys for reports are defined to identify uniquely each report. Special keys are defined to identify a set of special records which contain the extended keys, as defined above.

The support software reads or retains in memory the extended keys while a RDB file is open for access. In this way, operations which require access to extended key type information may be performed without the need for input/output. Facilities are supported to enable combinations of conditions with respect to key information to be generated, then used to access only those reports which satisfy those conditions.

Where whole files of data are required (eg. extraction for data assimilation) the key information may be ignored, and the file copied using the faster sequential form of access.

3.2 UNIX Version

The UNIX directory structure is used to represent the data base and reports type levels as described above. Since this operating system does not normally support indexed sequential access, the reports time level is comprised of two files for each valid time range - one containing the reports, the other an index file containing the extended keys.

The extended keys are further extended to contain byte location information for the corresponding report within the file of reports.

In this case, the support software includes routines which can read a given number of bytes beginning at a specific location within a file. As before, the extended keys are read or retained in memory to enable search operations to be optimised.

Where whole files of data are required, since the reports themselves are physically separated from the index file they can be used directly, or copied if necessary.

The data representation form chosen to meet the second requirement is FM 92 GRIB; products from other centres which have been pre-processed and converted to this form have also been merged to produce fields covering as large an area as possible; the structure used to retain fields on-line is called a fields data base (FDB).

5. FIELDS DATA BASE DESIGN

5.1 Conceptual Model

A three level model (see fig. 2.1) is used to define the major aspects of the data base design; the following principles influence this choice:

- a field, as defined above, is the basic unit of access;
- fields may be grouped by the verification time for which they are valid, qualified if necessary by the generating application (eg analysis, initialised analysis, forecast based on specific initial data, etc.) providing a typical unit of collective access; such a collection can be likened conceptually to a file containing the appropriate fields;
- such collections, or files may be grouped according to the system from which they were generated (eg. operations, operations test system, research experiment, generating centre, etc.);

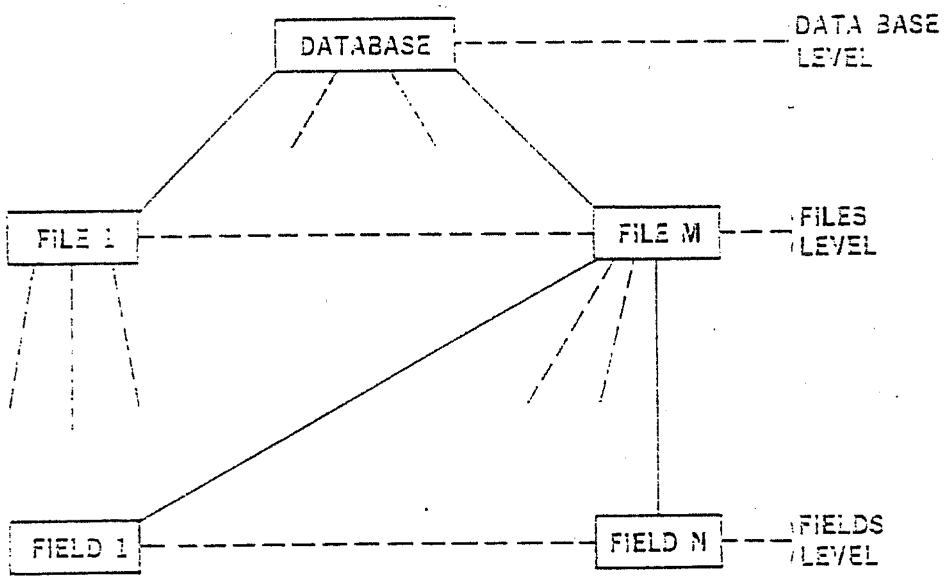

Fig. 5.1 Three level conceptual model

Thus, conceptually the fields data bases may be considered to have data base, file and field levels.

5.2 General Conceptual Design

At the DATA BASE level:

- one data base is required for each major implementation (operational system, test system, etc.)
- data base storage and access is supported by a single software package;
- write permission is restricted to selected, validated processes.

At the FILE level:

- fields for each verification time are mapped into separate files;
- distinction between generating application (forecast, analysis, etc.) with respect to the same verification time is achieved at the file level;

At the FIELDS level:

- each field is identified according by parameter, level type, and level;
- the field is the smallest unit which may be accessed.

5.3 Data Management Consequences

The above conceptual design requires that the products be organised in such a way that the smallest addressable object is a field, and that there be access to index type information indicating which fields are potentially available.

The products generation applications require fast access to the global fields resulting from the forecast system, coupled with the ability to store the generated products in such a way that they can subsequently be combined to enable all products for a given verification time to be passed to the dissemination sub-system. The conceptual design satisfies these needs.

Such a solution is also effective with respect to the pre-processing. There exists one pre-processing task for each data type within the message data base; the concept of one message type being mapped onto a set of reports types thus results in each pre-processing task reading from, and writing to files which are distinct from those addressed by other pre-processing tasks.

It is both desirable and necessary that individual fields be identifiable and accessible. This is achieved by the naming structures used at the data base, file, and field levels.

Not all research experiments generate the same set of products; it is thus important that there be a mechanism to identify which data bases are in being, and what each contains in terms of files and fields. It is also necessary to remove data no longer required, or which are time expired, and to support a back-up of important operational data; the data are large in volume, so the management of their retention must be effective and efficient.

Thus, the data management requirement of the physical implementation of a Fields Data Base is:

- fields, grouped in conceptual files according to verification time and generating application;
- information, to identify each field;
- access software.
- management software.

6. IMPLEMENTATION

6.1 Introduction

The current FDB implementation replaced an earlier system when the ECMWF Cray computer system moved to the UNICOS operating system. Many lessons learned from the previous version have been taken noted. In particular, there were considerable problems relating to the maintenance of index information with the previous version which needed to be avoided. The approach used was to utilise, where feasible, the de facto standard features of UNIX, and to plan for future extensions which would allow access to data within a Network File System from any UNIX platform.

6.2 Implementation strategy

The physical implementation follows closely the conceptual model. The data base level is a master directory for the data base. The files level is a sub-directory for each conceptual file. The fields level is a file for each field.

In order to make as much use as possible of the facilities offered by the operating system, naming conventions are