A Layman's Guide to TPFDF

Authors	:	Ruud Schelvis & Simin Marsden
Original Version	:	1.0, November 2002
Last Update	:	1.1, 29/11/2002
Issued by	:	Ruud Schelvis
Reviewed by	:	Geoff Lowry

1 Introduction 1
2 What is TPFDF? 2
2.1 History 2
2.2 Benefits of TPFDF 4
2.2.1 Application Programmer Productivity Benefits 4
2.2.2 System Management Benefits 5
2.3 Files and Subfiles 5
3 Use of TPFDF in the development cycle 7
4 Epilogue 9
5 List of acronyms 10

1 Introduction

This booklet aims to point out to Transaction Processing Facility (TPF) or Airline Control System (ALCS) professionals the considerable advantages of using the TPF Database Facility (TPFDF) product. For simplicity we shall just refer to TPF in this document and recognise that ALCS is included in that.

There are an ever-increasing number of databases in the TPF environment. The business logic operating on those databases is getting more and more complex and more and more data is required to support that business logic. In traditional TPF database handling there is no standard database organization. Therefore, there are no common routines for data retrieval, searches, sorts or updates. Application Programmers need to be intimately aware of the physical size or the database records and the location of data within them. TPFDF enforces a standard for database organization.

In the early days of the product series ACPDB, TPFDB and TPFDF there was huge resistance in traditional installations to the use of the database facility. Since the usage of the tool implies that additional overhead is created (the database access is not performed by the application but an additional layer is created) the fear was that the system performance would diminish. Particularly in the large American systems with a high system load, technicians were afraid to use it. As always, a change from a traditional low level Assembler approach to a higher level logical facility requires an intellectual leap. This is difficult, particularly for the older, more experienced people. The overhead is still there; the introduction of CEP (Common Entry Point) reduces overhead by removing the many ATTAC/DETAC operations that must be performed with the older "fast-link" mechanism.

There will probably always be traditional databases; although many installations have already done or have decided to start a migration process of traditional databases into TPFDF databases. This may seem costly, but in terms of future development flexibility and improved productivity, they have and/or will benefit from it! The major European airline installations spearheaded this process and some have been working on it for 15 years or more.

The following is a small comparison between TPFDF and 'traditional' databases from the point of view of the programmer. When programmers design a new application using a database, they ask several questions. Questions like "What if the inserted item causes the database to overflow", "What if the deletion of the item causes the database to become empty", "What if the items must be in a particular sequence for rapid retrieval" or "What if the database has multiple indexes". These scenarios can very quickly cause the program to be 80% database handling and only 20% "solving the business problem", unless TPFDF is used. Then the 80% database handling complexity disappears.

2 What is TPFDF?

The TPF Database Facility (TPFDF) is a licensed IBM product. It is a database manager for applications that run in a TPF or ALCS operating environment. Basically, TPFDF is a common interface between the applications and the database. The following figure shows the relative position of the TPFDF product in a TPF operating environment.

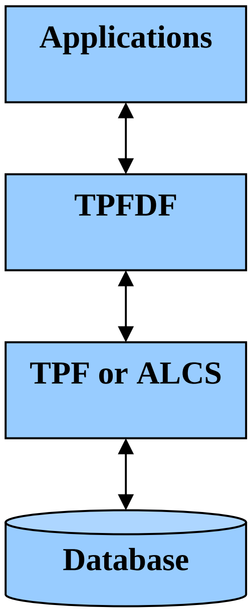

Figure 1 TPFDF System Overview

2.1 History

TPFDF is the current database product from IBM for the TPF and ALCS environments. It came into being as a result of a development partnership in the second half of the 1980s between IBM's International Airline Support Centre in the UK and the then Swissair Information Systems division in Zurich, now part of EDS.

The product was originally conceived and developed by Hans Eisele in 1981. Hans Eisele, one of the "Grand Old Men" of TPF was working for Swissair IS at the time and named the product ACPDB. In the years thereafter, the product evolved firstly into TPFDB with IBM's support and backing and is now the official IBM TPFDF offering as an option in the TPF product family. Enhancements to the TPFDF product are distributed separately from the base TPF products, but use the same PUT mechanism.

Today the product has become the de facto standard in the TPF high performance transaction database environments. The basic concepts and functionality designed in the early 1980s for the databases of the time still applies to the databases being developed two decades later. Of course, in the intervening time the product has developed a much wider range of functions, but it is still based on the initial concepts which proved to be the right design balance between flexibility and performance.

In the original TPFDF versions the programmer still had to specify the data levels that TPFDF could use, but this is no longer a requirement; **indeed this is no longer recommended and should be avoided.** Currently, TPFDF is able to allocate levels dynamically (within levels 9 - D). The latest development by IBM, DECB Support, even removes this level limitation as it provides a variable number of extra levels which can be created dynamically.

Data macro handling has also changed. When TPFDF first came on the market (in the form of ACPDB), all file characteristics had to be specified in the data macro. Now most file characteristics can be described in database tables, though data macro specification is of course still supported.

With the availability of **ALCS APAR AQ42082**, and **TPF Put 14**, TPFDF now provides a “**Common Entry Point (CEP)**” for the TPFDF interface with application programs. Previously access to TPFDF functions was through the use of a “**fast-link**” mechanism that caused the application program to branch directly to the TPFDF segment to perform the required function (DBADD, DBDEL, etc.). The provision of a “**Common Entry Point (CEP)**” allows for new facilities to be introduced (macro trace, for example) and reduces the complexity of the TPFDF interface. This may also become useful in debugging.

2.2 Benefits of TPFDF

TPFDF provides benefits over ‘traditional’ database management. First there is the application programmer productivity benefit and secondly there is the system management benefit.

2.2.1 Application Programmer Productivity Benefits

In traditional database handling the application programmer is responsible and must be totally aware of the database organization and structure. They need to be aware of the physical characteristics (size and location) of the data. They have to build software to access and modify the database. They have to ensure that overflow handling and indexing is correct. Although other applications may already manipulate the traditional databases, the handling of the database for the ‘new’ application still needs to be written and requires extensive testing.

Using TPFDF, the application programmer does not need to have database awareness. The database structure is totally removed from the programmer’s responsibility. All the programmer needs to do is focus on solving the business problem. This allows for the structure of the database to be changed at a later stage, perhaps to allow for increased number of ordinals or records, or perhaps for performance reasons, most often without requiring any changes to the application programs. Basically, TPFDF separates the database design from the application logic. The Database Administrator can then design the most appropriate database for the application, while the programmer needs only be concerned with writing the business logic into their program.

The TPFDF product provides high-level macros or C language functions that act as an interface between the application and the database. TPFDF enforces a standard for database organization and provides common routines for database access (find, modify, sort, merge, etc.). These common routines have proven to work properly, thus saving the application programmer time required to build and test his/her own database handling software. In addition, the application programmer only needs to know the logical relationships and structure of the data, not the physical characteristics. Since TPFDF applications are independent of the physical database, they are easier to enhance for new functional requirements.

When there is a need to add new data to an existing traditional database for whatever reason, it often happens that the layout of the record does not have any space for such enhancements. Traditional databases always seem to be full. The need to store additional data in an existing traditional database is often expensive as the applications, both existing and new, have to be rewritten to reconstruct the database such that ‘spare’ space is created. With TPFDF this is much easier.

What you often see is that a traditional database is expanded with a file-address reference to a TPFDF file. Or that a traditional file is gradually migrated to TPFDF. An example is the PNR record where the Special Service Requests (SSR) are stored in an additional TPFDF file linked to the PNR.

If you have ever attempted to add data to a traditional database when you are testing your application, then you will know how difficult and time-consuming it is. The TPFDF product has standard, well-proven tools available to add, replace or delete logical items and/or records.

2.2.2 System Management Benefits

The TPFDF product enforces a centralized database structure. Because the definition of the database is centrally maintained, certain database characteristics can be modified without affecting application programs. For instance, block size changes and any other physical databases changes are transparent to the application.

Various utilities are available to support the database administrator, like database integrity checks and data collection tools for monitoring performance. The data collection tool can highlight database designs that can cause performance problems. Simple modifications to the central database definitions may improve the performance of many application functions without a need to modify the programs themselves.

Another benefit is the 'on-line data repair'. For example, if you have a corrupted traditional record it usually takes some careful 'ZAFILing' to correct it. These types of actions are a lot easier under TPFDF with on-line entries to manipulate the files (just ask any helpdesk worker!).

A significant advantage using TPFDF is the high data integrity that is obtained from the standardised method in inserting, deleting and accessing records. This significantly reduces the possibility of database corruption or loss of data.

2.3 Files and Subfiles

A TPFDF database consists of files. Each file contains one or more subfiles. Each subfile contains a prime block and possibly one or more overflow blocks. These blocks contain logical records (LRecs). LRecs contain the actual data stored in the database.

TPFDF allows you to read LRecs from, and add LRecs to, a subfile in any file in the database, without having to worry about the physical structure of the file. You do need to know, however, how the file is split into subfiles, and what type of index support (if any) is being used with the file.

In a file, LRecs are distributed and accessed as follows:

- Algorithms
- Basic Index
- Block Index
- B+Tree Index

Actually, TPFDF files are not much different from traditional files. The difference is that the handling of the database access is transparent to the application. The application does not depend on the storage structure of the access path to the data. The actual steps involved in retrieving or building an indexed file are done automatically by TPFDF.

Like traditional records, TPFDF files can have various block sizes (for TPF block sizes L1, L2 and L4 are supported, for ALCS block sizes L1 to L8 are supported depending on the ALCS installation). For TPFDF, like traditional files, there are 3 types of physical files: Fixed Files, Miscellaneous Files and Pool Files.

3 Use of TPFDF in the development cycle

The following lists the 'typical' steps used in the development cycle:

1. Requirements specification
2. Functional Design (Logical Database and Application Design)
3. Technical Design (Physical Database and Application Design)
4. Application Build and Test
5. Implementation
6. Maintenance
7. Future enhancements

The initial benefits of TPFDF are the most obvious in the Application Build and Test plus Implementation phases. The first 3 steps will probably take the same amount of time irrespective of the choice between TPFDF and a traditional database. However, it is highly recommended to use TPFDF for new databases and to consider migration to TPFDF when enhancements to existing traditional databases are involved. Doing so radically improves productivity in steps 4 through 7. In particular, the amount of effort which ultimately goes into steps 6 and 7 over the lifetime of a TPF application usually vastly exceeds the total of steps 1 through 5. Such applications usually have a lifetime of decades.

Step 3 does not benefit from TPFDF but is very important as the benefits in the following steps depend on a good design. The following gives an overview of some design considerations.

3-Technical Design: A number of details must be taken into consideration when designing a TPFDF database. These details can make a big difference to performance issues once the database is established, and may become difficult to change. Hence it is better to consider such issues at the design stage.

- How much data will each subfile hold on average? This will involve considering the size of the prime block and whether over-flow blocks need to be of a different size.
- How will the data be retrieved? Do you need to retrieve based on a date, ID number, or name? This will determine what algorithm to use.
- How will the data be organised? Does the data need to be sorted or not? An organised database improves performance on queries, but if such queries are not required (for example a simple logging database), then the database need not be organised and thus save on the overhead involved in sorting the records.

The TPFDF advantages and features applicable for steps 4 through 7 can be summarized as follows:

4-Application Build and Test: TPFDF increases the programmer productivity. The application programmer only needs to know the logical relationships and structure of the data, not the physical characteristics. He/she does not need to worry about chaining and overflow. TPFDF utility commands (ZUDFM create/display/add/delete/replace etc.) facilitate the creation and manipulation of test data. ZUDFM OAS command provides formatted file information during trace.

5-Implementation: TPFDF utility commands can be used to initialise new files. TPFDF recoup can be used to validate the new database. TPFDF recoup uses information contained in database definition (DBDEF) tables to chain chase long-term pool file records that are defined using the DBDEF macro. TPFDF capture/restore utility, information and statistics environment (CRUISE) commands can be used for fallback scenarios during implementation. CRUISE is a validation, capture, and recovery tool for database administrators and system programmers and is based on file information that is stored in the TPFDF database definition (DBDEF).

6-Maintenance: TPFDF utility commands can be used to repair damaged files. DBDEF table provides online information for each file. Individual sub-files can be located online using the TPFDF utility commands by use of algorithm, ordinal, part, interleave and path arguments. TPFDF recoup can be used for database validation and CRUISE can be used for database management. TPFDF data collection commands can be used to display statistics relating to system use.

7-Future Enhancements: As TPFDF applications are independent of the physical database, they are easier to enhance for new functional requirements. A new type of Logical Record can be added to an existing database without disturbing the existing applications. Size of an existing Logical Record can be changed with minimum impact on existing applications. TPFDF recoup can be used to validate both existing and new databases after an enhancement. Should a problem occur during the implementation of an enhancement, CRUISE can be used to fallback to the old database structure.

4 Epilogue

We hope we have been able to clarify the basic characteristics, strengths and weaknesses of TPFDF and indicate the reasons to use TPFDF.

Interesting web sites for people who want to know more about TPFDF:

http://www-3.ibm.com/software/ts/tpf/pages/tpfdf.htm

Datalex provides education services to TPF and ALCS professionals. See the following link for the possibilities:

http://www.datalex.com/pdfs/Training_Todays_Top_.pdf
http://www.book-smart.com/TPF_Education/

5 List of acronyms

ALCS	Airline Control System ALCS is also referred to as TPF/MVS. Unlike TPF, ALCS runs as an application under MVS. As such it is able to utilize all the generic services that are provided to all jobs and applications running in this same environment.
CEP	Common Entry Point
CRUISE	Capture / Restore Utility, Information and Statistics Environment CRUISE is a validation, capture, and recovery tool for database administrators and system programmers.
LRec	Logical Record The internal format of individual TPFDF files does not correspond to the internal formats of records in a traditional TPF or ALCS database. In every TPFDF file, the data is organized into logical groups called logical records (LRecs). An LRec is the smallest unit of data that an application program can read, add, or delete.
PUT	Program Update Tape IBM distributes upgrades to the TPF and ALCS systems on a regular basis.
TPF	Transaction Processing Facility Try to obtain a copy of the 'Layman's Guide to TPF'. This will give you all the insight that you need.
TPFDF	TPF Database Facility Read this document to find out.