SDC-93-439

SDC
SOLENOIDAL DETECTOR NOTES

PROPOSAL FOR SDC DATABASE (PRELIMINARY)

February 8, 1993

Z.Y. Feng and T. Zhao
University of Washington

SDC-93-439

Proposal for SDC Database (Preliminary)

Z.Y. FENG and T. ZHAO
University of Washington
February 8, 1993

◇ ◇ ◇ ◇ ◇

1 Introduction

The importance of database for a HEP experiment as large and as complex as the SDC is selfevident. The SDC simulation software development and detector hardware design are progressing rapidly worldwide. A standardized database will aid the development of the detector simulation software and the detector hardware design. Eventually the database will also be an essential part of the analysis software.

A database for SDC will include three essential ingredients:

- detector geometric constants;
- a complete description of detector configurations and all GEANT related constants;
- calibration data.

The database should be unique for all SDC software, including data acquisition, Monte Carlo simulation, online and offline computations.

We have made a attempt to develop a database to meet the needs of the SDC experiment. The barrel muon drift tube system has been used as a model for the database development. The management system of this database, the data bank format, data input and output are discussed in this note.

2 USF - a New Data Management System

Data management (DM) systems at different levels have been developed and used in previous and recent HEP experiments. The most important ones are ZEBRA (CERN) [2], BOS (DESY) [3], JAZELLE (SLAC) [4], ADAMO (CERN/ALEPH and DESY/ZEUS) [5], DATMAN (DESY/H1) [6]. Among them, ZEBRA and BOS are basic level DM and the others are higher level DM. The higher level DM's are based on the basic level DM and they use the DDL (data definition language) to establish entity-relationships. The basic level DM systems have been playing a fundamental role for HEP experiment database. Only a few experiments have been chosen the higher level DM as their database tool.

For SDC, the basic level DM is now in urgent need from our opinion. The high level DM can be developed based on the basic level DM at a later stage of the software development. In fact, the database management system we are proposing will lead naturally to a higher level DM system in which the entity-relationship can be incorporated.

In our opinion, a good database must satisfy the following requirements:

- The organization of the database should be simple;
- The meaning of each attribute in the database should be obvious and clear;
- Creating, modifying and using the database should be easy even for a non-expert;
- Reading the database should be fast;
- The memory and disk space used by the database should be small;
- The DM system itself should be simple, easy to use and powerful.

Here, we propose a new basic DM system USF (**Universal, Simple and Fast data management system**) [7] for SDC database. The basic ideas of the package is originated from BOS and ZEBRA. All the code in USF is completely developed by Z.Y. Feng and bears no direct relation with any DM system mentioned above.

Several new features which BOS and ZEBRA do not have are incorporated in USF. These new features make USF more versatile.

3 Three Database Format

USF defines three data format. They are: TEXT, BINA and WORK. Their relationships are shown in **Fig. 1**.

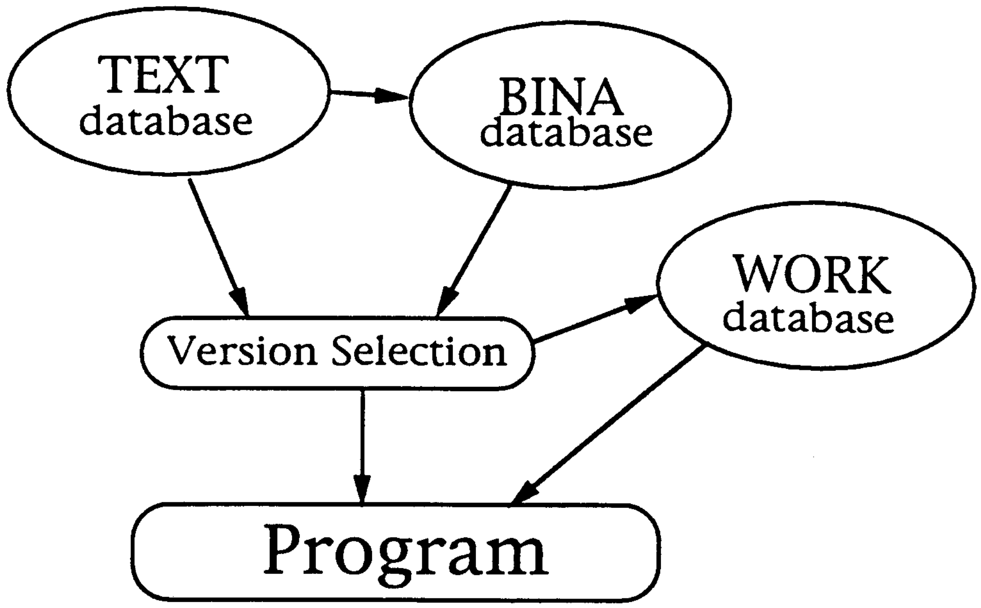

Fig. 1 Relationship among the three database format.

- TEXT:

  The database is created as blocks of text. Such form of database can be easily created and modified at any time during the software development. Comments can be inserted in order to make the meaning of each attribute obvious and clear. Another main advantage of TEXT format is that it can be correctly transfered between different computer systems via telnet and e-mail. USF allows the data banks in the TEXT database to have a number of different versions.

- BINA:

  Binary database is necessary for fast data input and output. The binary database is called BINA which is simply a copy of the text database in binary format. The binary database occupies much less disk space and can be much fast to access.

- WORK:

  WORK is a reorganized binary database with only one version for each bank. It has the same structure as USF's working arrays. These arrays are temporary buffers during running. They can be filled, dropped and cleaned any time for fast access and for saving memory.

  The format of WORK is designed in such a way that it can be used also for online and MC event input/output and reconstruction input/output.

Only a single FORTRAN call is needed to convert the database of different format, (either from TEXT to BINA and WORK, or from BINA to WORK) or import data to a program (from TEXT, BINA, WORK).

4 Rules of Creating a TEXT Bank

As in most of other DM systems, the basic cell of a datadase is a BANK. The rules for making a general database TEXT bank are:

1. A bank should have a name, which is the first word of the bank, with an 1-8 character string.
2. A bank should have an end mark "END$", which is the last word of the bank.
3. A bank should have a version stamp, which is the second word of the bank, with an 1-8 character string.
4. Between the version stamp and the end mark "END$", there can be any number of attributes (integers, reals and character strings).
5. Attributes:

   - String : the first character should be A-Z or a-z, other characters, except "!" "'", are all accepted as followings. But, only the first 4 characters will be counted;
   - Real : as FORTRAN rule, but without space in it;
   - Integer: as FORTRAN rule.

6. In a line, any characters after '!' will not be counted.
7. Attributes should be separated by at least one space key.

A example of a general TEXT bank is shown in **Appendix A**.

However, there are some exceptions for some control banks (see section 7 and examples in **Appendix B**. The rules for making those banks (TEXT) are:

1. A bank should have a name, which is the first word of the bank, with an 1-8 characters string.
2. A bank should have an end mark "END$", which is the last word of the bank.
3. Between the bank name and the end mark "END$", there can be any number of character string, each string may have a maximum of 80 characters.
4. A space and a "'" are also considered as characters, when they have a character of A-Z or a-z as the first character. So, ";" is used as the separator between two strings, and "$" is the end-mark of a sentence.
5. In a line, any characters after '!' will not be counted.

5 Bank Naming

We feel it will be better that the bank names should be obvious and meaningful. For example, we can name all of the banks related to the barrel muon detector BW1 as that the first three characters are "BW1". A maximum of eight characters is allowed as a bank name for recent version of USF, and is possible to have a longer bank name if desired. Characters in lower case and upper case are distinguished in USF. But in order to avoid confusions, we suggest that the upper case bank names should be always used.

Two specific names have been reserved for database file definition and bank version definition, they are "DATABASE" and "BKVS".

6 Version Naming

Having an active version stamp for a database bank in USF is a different feature from BOS and ZEBRA. We can have and use several versions of a bank in one database file. The version name is a string with 1-8 characters. For example, it can be "V01.1", "V930210" and so on. We should decide a naming rule to allow accessing a bank of the latest version by default.

Using an integer or real number for version stamp is also allowed by USF. It depends on users taste.

7 Two Special Banks and Their Use

Two special banks are defined:

1. Bank "DATABASE", which defines the file name and format of a database.
2. Bank "BKVS", which defines the versions of banks.

Appendix B gives an example of these two banks, in which BKVS defines the versions of BW's banks.

When using USF, it first reads automatically the DATABASE bank and BKVS bank from a special file data, for example, SDCSTRG.DATA, and then it opens the database file, and finally it reads all of database banks into the working buffer for late use.

8 Getting a Bank

Getting a bank from the working buffer is done by a single calling

```
CALL GETBNK(NAME)
```

where, NAME is the bank name. The data of the bank will be stored in the common block BS (**Appendix C**), where NBS indicates how many words have been stored. The result of this call can be:

1. The bank and the specific version through BKVS exists. The data stored in BS will be just what you wanted and the output NBS will be greater than 0;
2. The bank exists, but you did not give a version name through BKVS or the given version name does not exist. The data stored in BS will be the data of the default version and the output NBS will be greater than 0. If the given version name doesn't exist, a warning message will be given;
3. If the bank does not exist, NBS will be zero and a warning message will be given.

Appendix D is an example subroutine which shows how to use a database bank.

9 Structure of SDC Database

It is inconvenient to have all SDC database in one big TEXT file. Small TEXT files will be easier for modifying and for communicating via networks. The entire SDC database can be divided into detector geometry database, calibration database and maybe others.

The geometry database can be again divided into several subdatabases. Let's say three subdatabases, for example, GEOMTKDB (Tracking System), GEOMCLDB (Calorimeter System) and GEOMMUDB (Muon System), according to the hardware design. The calibration database can be also divided into several subdatabases, like the geometry subdatabases, for example, CALITKDB (Tracking System), CALICLDB (Calorimeter System) and CALIMUDB (Muon System). Each subdatabase can have several TEXT database files.

A number of TEXT files can be combined in to a BINAry database file by a single USF call. A user can select the TEXT files which are relevant to his/her analysis and assemble these files into a BINAry file. A user can also select any BINAry files to be converted into WORK format. This feature provides an easy access and flexibility.

The proposed SDC database structure is shown in **Fig. 2**.

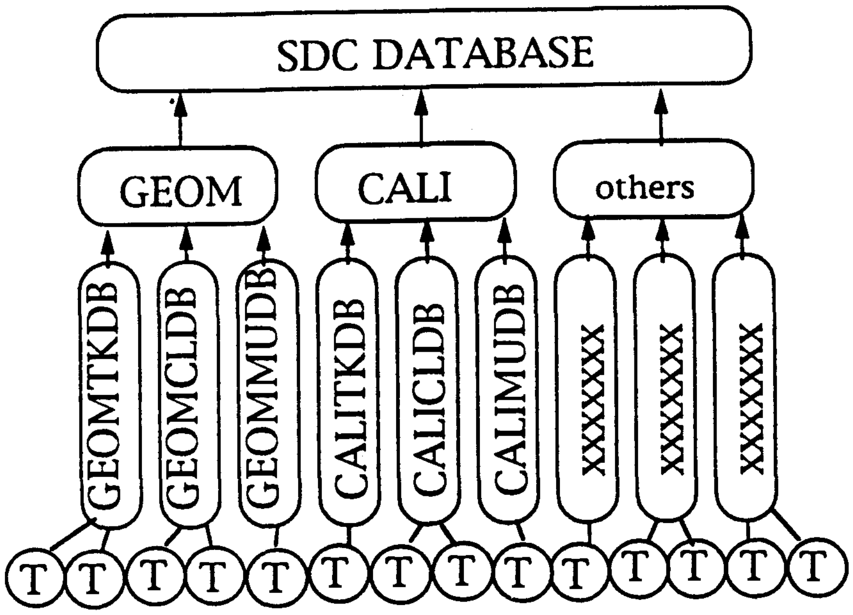

Fig. 2 Structure of Proposed SDC database

10 Practice

The ideas about the database have been partly practiced for E781 (FNAL) database design[8], where USF was used as the interface for inputing TEXT database into the General ADAMO File (GAF). The authors have fully tested these ideas with UNIX/(SGI,HP) and VMS/VAX systems for SDC muon system while making the mapping function (MUONMAP [9]), the trigger simulation package (MUONTRG [10]), and a fast Monte Carlo simulation package for SDC muon system.

The authors would like to thank H. Lubatti, V. Chaloupka, C. Daly, S. Kunori and A. Gauthier for their helps and discussions during the work.

References

1. GEANT3 User's Guide, W5013/W013-E, R. Brun et al., July 1991.
2. ZEBRA - Kernel Data Structure Management System, R. Brun and J. Zoll, Long Write-up Q100, CERN, Jan. 20, 1987.
3. The BOS system, V. Blobel, DESY R1-88-01.
4. JAZELLE Users Manual, A.S. Johnson, SLD Software note, SLAC, 1989.
5. The ADAMO Data System - An Introduction for Particle Physics, M.G. Green, RHBNC 89-01; The ADAMO Data System - Programmers Manual (Version 3.2), S.M. Fisher and P. Palazzi, June 1992.
6. Proposal for H1 Data Model (H1DM), Z.Y. Feng et al, H1-09/89-120, 1989. The H1 Data Management Tool Package for Reconstruction and Analysis, U. Berthon et al., H1-06/90-140, 1990.
7. USF - a Universal, Simple and Fast data management system, Z.Y. Feng, to be ready in March, 1993.
8. E781 Database and GEANT Application, Z.Y. Feng, distributed in E781 collaboration, Oct. 15, 1992.
9. Mapping Function for SDC Muon System (MUONMAP), Z.Y. Feng and T. Zhao, SDC-93-440, Feb. 10, 1993.
10. Trigger Simulation for SDC Muon System (MUONTRG), Z.Y. Feng and T. Zhao, SDC-93-441, Feb. 10, 1993.

Appendix A

An example text bank BW1S, data copied from C. Daly.

```
BW1S ! Bank name: BW1's sparsification bank for theta tubes
v01     ! version number
!
      2 ! Module Number
      1 ! Layer Number
6210.85 ! r, in mm
2552.62 ! half tube length (?), in mm
     91 ! Number of Tubes
! Sparsification in mm
-4234.5 -4140.4 -4046.3 -3952.2 -3858.1
-3764.0 -3669.9 -3575.8 -3481.7 -3387.6
-3293.5 -3199.4 -3105.3 -3011.2 -2917.1
-2823.0 -2728.9 -2634.8 -2540.7 -2446.6
-2352.5 -2258.4 -2164.3 -2070.2 -1976.1
-1882.0 -1787.9 -1693.8 -1599.7 -1505.6
-1411.5 -1317.4 -1223.3 -1129.2 -1035.1
-941.0 -846.9 -752.8 -658.7 -564.6
-470.5 -376.4 -282.3 -188.2 -94.1
0.0 94.1 188.2 282.3 376.4
470.5 564.6 658.7 752.8 846.9
941.0 1035.1 1129.2 1223.3 1317.4
1411.5 1505.6 1599.7 1693.8 1787.9
1882.0 1976.1 2070.2 2164.3 2258.4
2352.5 2446.6 2540.7 2634.8 2728.9
2823.0 2917.1 3011.2 3105.3 3199.4
3293.5 3387.6 3481.7 3575.8 3669.9
3764.0 3858.1 3952.2 4046.3 4140.4
4234.5
!
      2 ! Module Number
      2 ! Layer Number
6306.55 ! r
2592.26 ! half tube length (?)
     90 ! Number of Tubes
-4259.75 -4164.2 -4068.65 -3973.1 -3877.55
-3782.0 -3686.45 -3590.9 -3495.35 -3399.8
-3304.25 -3208.7 -3113.15 -3017.6 -2922.05
```

```
-2826.5 -2730.95 -2635.4 -2539.85 -2444.3
-2348.75 -2253.2 -2157.65 -2062.1 -1966.55
-1871.0 -1775.45 -1679.9 -1584.35 -1488.8
-1393.25 -1297.7 -1202.15 -1106.6 -1011.05
-915.499 -819.949 -724.4 -628.85 -533.3
-437.75 -342.2 -246.65 -151.1 -55.55
40.0 135.55 231.10 326.6498 422.1998
517.7497 613.2996 708.8496 804.3995 899.9495
995.4994 1091.049 1186.599 1282.149 1377.699
1473.249 1568.799 1664.349 1759.899 1855.449
1950.999 2046.549 2142.099 2237.649 2333.199
2428.749 2524.298 2619.848 2715.398 2810.948
2906.498 3002.048 3097.598 3193.148 3288.698
3384.248 3479.798 3575.348 3670.898 3766.448
3861.998 3957.548 4053.098 4148.647 4244.197
....
END$
!
```

Appendix B

An example of DATABASE bank and BKVS bank.

```
!
DATABASE
! Name Format
SDCMUDB.BINA; BINA;$
! SDCMUDB.TEXT; TEXT;$
! SDCMUDB.WORK; WORK;$
END$
!
!
BKVS ! bank name: this bank is for defining version of each bank.
! BANK name VERSion name
BW1SHP; v01;$        ! BW1's shape bank
BW2SHP; v01;$        ! BW2's shape bank
BW3SHP; v01;$        ! BW3's shape bank
BW1L; v01;$          ! BW1's location bank
BW2L; v01;$          ! BW2's location bank
BW3L; v01;$          ! BW3's location bank
BW1S; v01;$          ! BW1's sparsification bank for theta tubes
BW2S; v01;$          ! BW2's sparsification bank for theta tubes
BW3S; v01;$          ! BW3's sparsification bank for theta tubes
BW1P; v01;$          ! BW1's sparsification bank for phi tubes
BW3P; v01;$          ! BW3's sparsification bank for phi tubes
BW1PL; v01;$         ! BW1's tube length bank for phi tubes
BW3PL; v01;$         ! BW3's tube length bank for phi tubes
BW3THETA; v01;$      ! BW3's theta-stereo tube bank
BW3TPHI; v01;$       ! BW3's phi-stereo tube bank
END$
```

Appendix C

Common block to store a bank information:

```
+KEEP, BS.
      INTEGER       NMAX,NBS
      PARAMETER  (NMAX=10000)
      REAL          BS(NMAX)
      INTEGER       IBS(NMAX)
      CHARACTER*4  CBS(NMAX)
      COMMON  /FS/NBS,BS
      EQUIVALENCE (IBS(1),BS(1),CBS(1))
```

Appendix D

An example subroutine to get a bank.

```
C
+DECK, EXBW1S.
C
      SUBROUTINE EXBW1S
+SEQ,BS.
      DIMENSION BWS(95,4,3),NBWS(4,3),R(4,3),HF(4,3)
      NBS=0
      CALL GETBNK('BW1S')
      IF(NBS.LE.0)GOTO 11
      IND=0
1     CONTINUE
      M =IBS(IND+1)
      L =IBS(IND+2)
      R(L,M) = BS(IND+3)*0.1
      HF(L,M)= BS(IND+4)*0.1
      NBWS(L,M)=IBS(IND+5)
      IND=IND+5
      CALL UCOPY(BS(IND+1),BWS(1,L,M),NBWS(L,M))
      IND=IND+NBWS(L,M)
      IF(IND.GE.NBS)GOTO 10
      GOTO 1
10    CONTINUE
      ...
      ...
      RETURN
      END
C
```