IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661, p- ISSN: 2278-8727Volume 16, Issue 1, Ver. VI (Feb. 2014), PP 66-69
www.iosrjournals.org

Mapping Procedural Modules To Storage

Vijaylaxmi Bittal,Shruti Mittal
[1]*Department of Information Technology, RAIT, Nerul,Navi Mumbai*
[2]*Department of Information Technology, RAIT, Nerul,Navi Mumbai*

Abstract*: Majors in computer science accept structured programming as a subset of procedural programming paradigm and it is intended for complex system design and development. But as everyone is moving towards object orientation so as to model real world , which will be easily mapped to database So we also need to have a conversion tool for mapping procedural modules to storage. Our goal is to design a methodology to convert (map) structure/record to storage.*

Index Terms: *Structure, Record, procedural paradigm, storage.*

I. INTRODUCTION

A. Structured and procedural programming

Most often we should be able to use the term interchangeably but with subtle differences. When higher level languages begun to get richer, one realized that all units of work should be broken into smaller tractable parts. Structured programming [2] is any programming when functionality is divided into units like for loop, while loop, if... etc. Also, here the piece of code (function) can be re-used. In procedural programming, one can create a physical form of package which are fairly general purpose and re-usable. This is called modules of elements compiled together. So one can hardly see modular programs which are not structured and vice versa.

B. Object Oriented Programming

This is well defined in literature. Understand that [1]Object oriented programming is a form of structured programming by definition. So basically structured code where functions (or procedures) dominant over data is called procedural where as class and object based representation is called object oriented.

C. Storage

Database allocates logical space for all data in the database. The logical units of database space allocation are data blocks, extents, segments, and table spaces. At a physical level, the data is stored in data files on disk. A database is a set of files that store data in persistent disk storage.

1. Data files and temp files

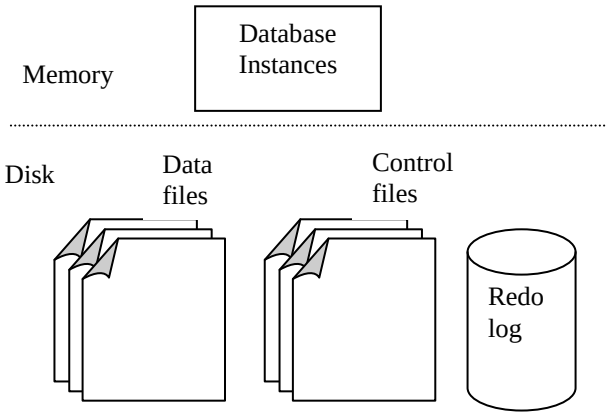

Fig 1.1 Database Instance and Database Files

A data file is a physical file on disk that was created by any of the conventional Database and contains data structures such as tables and indexes. A temp file is a data file that belongs to a temporary table space. The data is written to these files in an Oracle proprietary format that cannot be read by other programs.

2 .Control files

A control file is a root file that tracks the physical components of the database.

3. Online redo log files

The online redo log is a set of files containing records of changes made to data.

A database instance is a set of memory structures that manage database files. Figure 1.1 shows the relationship between the instance and the files that it manages.

One characteristic of an RDBMS is the independence of logical data structures such as tables, views, and indexes from physical storage structures. Because physical[3] and logical structures are separate, you can manage physical storage of data without affecting access to logical structures. For example, renaming a database file does not rename the tables stored in it. From this analogy we conclude that it's better to have mapping between procedural modules to storage (database).

II. Design

A. Data flow diagram

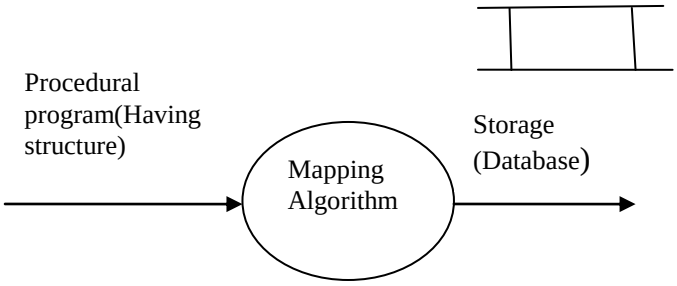

Fig 2.1 Context Diagram.

The Fig 2.1 shows the context diagram. In this procedural program (Containing structure) is an input and it is processed by Mapping Algorithm, which produces output as storage (database).

B.Design Steps

1. Read the input program(c program containing structure)
2. Map structure name as table name.
3. For each structure field map a column in the table with the name of the field in structure.
4. Then each structure variable corresponds to a record in the table.
5. When mapping fields , we need to map data type for the database column. (When mapping primitive types the correspondence between programming language type and the database type is direct. While dealing with non primitive data type , the type text in database requires a specified maximum size.
6. Next mapping a primary key. There are two options:
   i)Identify the structure field that uniquely locates particular record.
   ii)Add a unique identifier that we generate.
7. Repeat same steps(1-6) for next input.
8. Stop

C. Case Study

Now we are applying design steps to the case study.

```
#include<stdio.h>
struct  student
{
int  roll_num;
char name[30];
char branch[10];
int age;
char sex;
int marks;
};
void main( )
{
struct  student  s[10];
int n,i;
printf (“enter the no of students”);
scanf(“%d”,&n);
for (i=1; i<=n; i++)
```

```
{
printf ("enter details of  student %d \n ", i);

printf("enter the roll_num \n");
scanf("%d",&s[i].roll_num);

printf("enter name \n")
scanf("%s",&s[i].name);
printf ("enter the branch \n");
scanf("%s ",&s[i].branch);

printf ("enter the  sex \n");
scanf("%c",&s[i].sex);

printf ("enter the total marks \n");
scanf("%d ",&s[i].marks);
}
getch( );
}
```

The above program contains structure[4] having student information. Using structure variable we are accessing the fields of structure and displaying the details of corresponding students.

Now we are applying our Design Concept to the above program that is going to map this structure to storage(database).

1. Read the input program(c program containing structure).
2. Map structure name as table name.
   student – > table name

Student

3. For each structure field map a column in the table with the name of the field in structure.

Fields are –

roll_num, name, sex , branch, marks

roll_num	name	sex	branch	marks

4. Then each structure variable(s[1],s[2]…….s[n]) corresponds to a record in the table.

For example,

roll_num -1, name-ABC, sex- F, branch- IT, marks - 650

roll_num	name	sex	branch	marks

5. When mapping fields , we need to map data type for the database column. (When mapping primitive types the correspondence between programming language type and the database type is direct. While dealing with non primitive data type , the type text in database requires a specified maximum size).

roll_num – int(primitive)
name- string(name[30])(non-primitive)
sex- char(primitive)
branch- String(branch[5])(non- primitive)
marks – int(primitive)

roll_num	name[30]	sex	branch[5]	marks

6. Next mapping a primary key. There are two option are there , - Identify the structure field that uniquely locates particular record.In this case study we identify the roll_num is unique field that we choose as primary key for student table.

roll_num(primary key)	name[30]	sex	branch[5]	marks

7. Repeat same steps(1-6) for next input.
8. Stop.

Suppose we re taking below example of nested structure the Design steps would be same with minor changes.

```
struct  invoice
{
  char inm[15];
  int qty;
  int rate;
  int amt;
    struct date
    {
      int dd;
      int mm;
      int yy;
    } d;
 } l [100];
```

Since there are two structures two tables (Storage structures) will be created
Outer structure name will be the name of first table.
For our example Table name-> invoice.
Inner structure name will be the name of Second table.
For our example Table name-> date
Each field of the outer structure corresponds to a column in the First table
Each field of the inner structure Corresponds to a Column in the second table
If we need to merge all the Information of both the structure this can be implemented by Foreign key, like by using ampersand (&)structure variable of outer Structure dot operator(.) variable of inner structure dot operator(.)Field. We can access the records of Second table.

The nesting can be performed up to any level.

III. Conclusion

In this paper we concluded that our methodology for mapping procedural modules (using structure) to storage (database) is beneficial for organizing data. And we have proved the same methodology with case study(C program).

This work can be extended to perform other database queries.

References

[1]. Su Jian; Weng Wenyong;Wang Zebing "A Teaching path for Java object oriented Programming" Information Technology and Applications 2009
[2] Whity R.W; Fenton, N.E; Kapsi, A.A,"Structured Programming: A tutorial Guide" Volume 3, Issue 3, 1984
[3] Dewan, RM; Gavish B"Modules for the combined logical and physical design of the database "Volume 38, Issue 7, 1989.Computers IEEE
[4] Pan S, Dromey R G "Structured Programming" Software Engineering 1996 18th International Conference