ABSTRACT

The use of relational database technology is growing rapidly. IBM's relational database, DB2, is widely used in large IBM mainframe installations. Systems developed using relational technology are primarily used for data reporting and analysis. These functions are normally performed by high level languages such as the SAS System. SAS/DB2 Software is the interface between the SAS System and DB2. Users should be aware of DB2 performance considerations prior to running a SAS/DB2 program to extract the data required for reporting and analysis. This paper will use SAS/DB2 software to examine DB2 performance issues.

1.0 INTRODUCTION

In order to use SAS/DB2 Software for reporting and analysis of data stored in IBM's relational database management system, DB2, the user must be familiar with DB2 itself, the SQL language which is used to communicate with DB2, and of course, SAS/DB2 Software. A brief overview of the structure and purpose of these subjects will be discussed.

1.1 DB2 Overview

Database 2, commonly referred to as DB2, is IBM's relational database management system, which runs on IBM mainframes under the MVS operating system. A DB2 database is a logical entity referring to a set of tablespaces. Tablespaces are physical spaces where tables are stored. Tables contain data stored in rows and columns like a flat file or a SAS data set. Rows can be correlated to records in a flat file or observations in a SAS data set. Columns can be correlated to fields in a flat file or variables in a SAS data set.

Thus, the user will see a DB2 database as a collection of tables. Each table contains data based on an entity or subject. The column or set of columns which uniquely define a row for a table is called the primary key for that table. A foreign key is a column or set of columns whose values are matched with the primary key in another table to join the two tables. Several tables can be joined in this manner to produce the data required for a given application.

The primary key of a DB2 table table is typically also an index. An index to a DB2 table allows data to be accessed directly rather than reading the table sequentially. This shortens the wait time for the user and saves computer time. A DB2 table will often have three or four indexes, each consisting of one or more columns. Indexes are assigned based on the most frequent way(s) that users need to extract data.

DB2 databases, tablespaces, tables, primary keys, foreign keys, and indexes are referred to as DB2 objects. There are other DB2 objects, but they are not relevant to the subject presented in this paper.

1.2 SQL Overview

Structured Query Language (SQL) is used to communicate with DB2, no matter what language, be that COBOL, SAS/DB2 Software, etc., is used in the application. The statements that comprise SQL are grouped into the following categories:

- Data Definition Language: includes the statements, CREATE, ALTER, and DROP, which perform utility-type functions on DB2 objects.
- Data Manipulation Language: includes the statements, SELECT, UPDATE, INSERT, and DELETE. The SELECT statement is used to extract data stored in DB2 tables and is the most frequently used SQL statement. UPDATE, INSERT, and DELETE statements are used to make changes to the data stored in a DB2 table.
- Data Control: includes the statements, COMMIT, LOCK, and ROLLBACK, which are typically used only by the DB2 Database Administrator to control the updating process.
- Security: includes the statements, GRANT and REVOKE, which control user access.
- Third Generation Access: includes several statements which are used in conjunction with languages such as COBOL in order to access DB2 databases.
- Other: The only statement in this category, which is of interest in this paper, is the EXPLAIN statement. The EXPLAIN statement is used to enable the user to examine the data access path selected by the DB2 Optimizer prior to running the program.

1.3 SAS/DB2 Software Overview

SAS/DB2 Software executes SQL statements through the use of three procedures:

PROC DB2EXT
PROC DB2EXT is used to extract data stored in DB2 tables and place them in a SAS data set. The SQL SELECT statement is executed by PROC DB2EXT to perform this function.

PROC DB2UTIL
PROC DB2UTIL is used to make changes to a DB2 table, including adding rows to a DB2 table, updating column values in a DB2 table, and adding new columns to a DB2 table. The SQL INSERT statement is used to add data to a DB2 table. The SQL UPDATE statement is used to update column values, and the SQL ALTER and UPDATE statements are used to add a new column. The SQL COMMIT statement is along with these SQL statement to save the changes. In addition, PROC DB2UTIL can be used to execute any SQL statement, other than the SELECT statement.
2.0 PROC DB2EXT EXAMPLE 1

PROC DB2EXT is the primary SAS/DB2 Software procedure used because it performs the function of data extraction which is the starting point for data reporting and analysis. PROC DB2UTIL and PROC DB2LOAD perform functions that are not typically required by the user community. An example of the use of PROC DB2EXT will be shown in this section.

2.1 Database Layout

The DB2 database to be used in this example is depicted in the drawing shown below. The CLAIM table contains information such as the member number, id, age, and claim number. The MEMBER table contains information on the patient such as enrollment date, persons covered, and member address. The GROUP table contains information about the member's employer, such as the employer name, address, and what benefits are covered. The SERVICE table contains medical detail information on each service performed, such as the type of procedure, the diagnosis, the service dates, provider (hospital/doctor), the amount charged, coinsurance, deductible, and amount paid. Several decode tables are associated with the SERVICE table. Decode tables are used to convert columns stored as codes into English descriptions. The PROVIDER table contains information about the hospital or doctor who performed the service, such as provider name, address, and speciality (Ophthalmologist, Family Practice, etc.). The SPECIALITY decode table converts the speciality code into an English description.

2.2 Indexes, Keys, & Size

The example to be shown in this section will be based on the CLAIM, SERVICE, PROVIDER, and SPECIALITY tables. In order to use these tables, it is important to know their indexes, keys, and size. If the data is accessed using an index, the run time will be decreased. This is particularly important when there is a large number of rows. Therefore, it is also important to know the approximate size of the DB2 table. Knowing the primary key for each table will enable the user to better understand the data that will be extracted and he will be better equipped to perform data reporting and analysis. The following chart shows the indexes, primary keys, and approximate size of the DB2 tables, which will be used in the example to be shown in this section.

2.3 PROC DB2EXT Code

A report is needed which will show specific information from the CLAIM, SERVICE, PROVIDER, and SPECIALITY tables for a member whose id is 009094644. PROC DB2EXT is used with a SELECT statement requesting the desired columns. The FROM clause is used to name the DB2 table from which this data will be extracted. The single letter coded after each table name is used as a reference preceding the column name in the prior select list. This is only required if a column from more than one table has the same name. However, it is good program documentation to always use this reference. The WHERE clause of the SELECT statement is used to select the member id 009094644, and is also used to match the primary and foreign keys of the requested tables. Whenever data is requested from more than one DB2 table, this must be coded. If not, the resulting SAS data set will contain the CARTESIAN product. In other words, if two tables were joined which had 5 and 10 rows, the result would be 50 rows; all possible combinations of 5 and 10. In applications, you will typically want a subset of the CARTESIAN product, based on the matching of primary and foreign keys.

The data extracted from these four DB2 tables will be stored in a SAS data set named in the DATA= option of the PROC DB2EXT statement; in this case, CLAIMS. The RENAME statement is used to assign names to the SAS variables resulting from the DB2 columns selected. The SAS variable names are specified after a number which represents the order that the DB2 columns are listed in the SELECT statement. Without a RENAME statement, the SAS variables are named based on the first eight positions of the DB2 column name. The UNIQUENESS option of the PROC DB2EXT statement is used in order to generate unique SAS variable names when the first eight positions of the DB2 column name is the same for more than one column. The SSD option is installation specific and is the subsystem id.

Index Name | Index Column(s) | PROC DB2EXT OUT= CLAIMS UNIQUE SSDID=DSF;
------------|-----------------|-------------------------------------------------------------------
CLAIM (30,000,000 rows): | mem_id, clm_adju_date, sys_time | 1
ICLAIM * | mem_id, clm_adju_date | 2
ICLAIM | mem_id, clm_adju_date | 3
ICLAIM1 | serv_id, clm_adju_date | 4
ICLAIM2 | serv_id, clm_adju_date | 5
ICLAIM3 | serv_id, clm_adju_date | 6
SERVICE (66,000,000 rows): | mem_id, clm_adju_date, sys_time, serv_line | 7
ISERV * | mem_id, clm_adju_date, serv_line | 8
ISERV2 | serv_id, serv_prov_id | 9
ISERV3 | serv_id, serv_prov_id | 10
PROVIDER (35,000 rows): | prov_id, serv_prov_id | 11
IPROV * | prov_id, prov_ser_id, serv_prov_id | 12
IPROV2 | prov_speciality_code | 13
SPECIALITY (100 rows): | | 14
* Primary Key

1. PROC DB2LOAD

PROC DB2LOAD is used to create and load a DB2 table with data from a SAS data set. The SQL CREATE, INSERT, and COMMIT statements are used to perform this task. Like PROC DB2UTIL, PROC DB2LOAD can also be used to execute any SQL statement, other than the SELECT statement.
The composition of the SELECT statement is an important factor in the selection of the data access path. For example, if an index is used in the WHERE clause, the DB2 Optimizer will usually use the index to access the required rows. If an index is not used in the WHERE clause, the DB2 Optimizer will read the table sequentially (tablespace scan).

The data access path selected by the DB2 Optimizer is critical to the performance of the application, especially when large DB2 tables are involved. It is a real advantage for the user to know what access path has been selected prior to running the program. The PLAN TABLE is a DB2 table which contains information about the data access path selected by the DB2 Optimizer. The PLAN TABLE is created by the SQL EXPLAIN statement. This statement can be executed and the PLAN TABLE examined prior to running the program. If the data access path selected by the DB2 Optimizer is inappropriate, the user can alter the coding of the SELECT statement in order to get the DB2 Optimizer to select a different access path.

1.1 Allocate a PLAN TABLE

In order for the user to examine the data access path selected by the DB2 Optimizer, the first step is to create a PLAN TABLE. Each user will have their own PLAN TABLE. PROC DB2UTIL is used to execute two SQL statements. The first is a CREATE TABLESPACE statement, which creates the physical space to hold the PLAN TABLE. The tablespace name is PLNTBL, but another name can be selected. The tablespace is for TSID717, the user's id. You should consult your DB2 Database Administrator, especially for the appropriate set up of the CREATE TABLESPACE statement in your company. The second SQL statement executed is a CREATE TABLE statement, which creates the PLAN TABLE, along with the appropriate columns and their data attributes. The columns and their data attributes are fixed. The PLAN TABLE is usually created after the DB2 Optimizer has selected the data access path. If the data access path is selected by the DB2 Optimizer, the user can alter the coding of the SELECT statement in order to get the DB2 Optimizer to select a different access path.

2.4 PROC DB2EXT Results

The first few observations of the report generated by PROC PRINT, based on the SAS data set created by PROC DB2EXT, are shown below:

```
CLAIMS FOR 009094644

OBS CLMID PAIDDTE PROC ORIGCHG PAID PROVID PROVNAME PROVSPC
1 008205 880725 84182 10 MED LAB OF DR SMITH
2 008308 881104 82543 22.34 FRANK W HILL MD
3 008363 881229 90480 25.00 FRANK W HILL MD
4 008308 881104 36435 3.16 MED LAB OF DR SMITH
5 008279 881005 T1030 20.00 BEECH GROVE PODIATRY
6 008363 881229 90480 25.00 FRANK W HILL MD
7 008273 880930 90440 8.00 FRANK W HILL MD
8 008215 880802 90440 3.42 INDEPNT LAB
9 008205 880725 82543 22.34 MED LAB OF DR SMITH

NOTE: THE PROCEDURE D82EXT USED 2.34 SECONDS AND 1248K.
NOTE: THE PROCEDURE PRINT USED 0.08 SECONDS AND 912K AND PRINTED PAGES 1 TO 3.
NOTE: SAS USED 1248K MEMORY.
NOTE: SAS INSTITUTE INC.
SAS CIRCLE
PO BOX 8000
CARY, N.C. 27511-8000

2.5 DB2 UTIL Commands

DB2 UTIL creates the PLAN TABLE and keeps track of the columns and their data attributes. The columns and their data attributes are fixed. The PLAN TABLE is created in the tablespace TSID717.PLAN TABLE.

```
```
The SELECT statement used in the previous example.

4.0 SQL EXPLAIN STATEMENT

The SQL EXPLAIN statement is executed for each DESIRABLE SELECT statement and the result selected by the DB2 optimizer, the PLAN TABLE for every DB2 table used in the SELECT statement is put in a SAS data set named plan. PROC PRINT is then used to print the SAS data set, plan.

4.1 SQL EXPLAIN Statement Example 1

The SELECT statement used in the previous section 2 is used in the EXPLAIN statement shown below.

```sql
PROC DB2UTIL SSID=DSP;
NOTE: FIELDS REQUIRED FOR UPDATE WERE NOT ENTERED OR IN ERROR
NOTE: TABLES OPTION REQUIRED FOR UPDATE PROCESSING WAS NOT ENTERED
NOTE: THE PROCEDURE DB2EXT USED 0.42 SECONDS AND 1212K.
```

The contents of the PLAN TABLE are shown below. Two variables, PLANNO and ACCESSTY, are of particular interest. PLANNO indicates which table, denoted by variable TNAME, is accessed first. ACCESSTY is equal to I when the table will be read sequentially. In this case, the CLAIM table is accessed first by using the INDEX MATCHCOLS. The only difference is the SELECT statement to be executed is changed. This SELECT statement requested a specific claim id to be selected.

4.2 SQL EXPLAIN Statement Example 2

Every time the EXPLAIN statement is executed, a row is added for every table shown in the FROM clause of the SELECT statement. Rows from previous executions of the EXPLAIN statement are not replaced; new rows are added. Typically, the user does not need to save results from previous executions of the EXPLAIN statement. The first step in the following program was PROC DB2UTIL to execute the SQL DELETE statement. This deletes all rows in the PLAN TABLE, but does not actually delete the table itself. The remainder of this program is like the program shown in the previous section. The only difference is the SELECT statement to be explained has changed. This SELECT statement requested a specific claim id to be selected.
EXPLAIN PLAN SET QUERYNO=1
FOR SELECT STATEMENT;
FOR TABLE = OPTION REQUIRED FOR FIELDS REQUIRED FOR UPDATE
WHERE C.MEMB_ID = S.MEMB_ID
AND C.CLM_ADJU_DATE=S.CLM_ADJU_DATE
AND C.SYS_TIME=S.SYS_TIME
AND S.SERV_PROV_ID = P.PROV_ID
AND S.SERV_PROV_SUFX_ID = T.PROV_SUFX_ID
AND P.PROV_SPEC_CODE = T.PROV_SPEC_CODE
AND C.FST_6_CLKM_ID = '008363';

NOTE: FIELDS REQUIRED FOR UPDATE WERE NOT ENTERED OR IN ERROR
NOTE: TABLE= OPTION REQUIRED FOR PROCESSING WAS NOT ENTERED
NOTE: THE PROCEDURE DB2UTL USED 0.43 SECONDS AND 1212K.

PROC DB2UTL OUT=PLAN UNIQUE SSID=DSP;
SELECT * FROM TSID717.PLAN_TABLE;
TITLE! 'EXPLAIN EXAMPLE 2';
NOTE: THE DATA SET WORK.PLAN HAS 4 OBSERVATIONS AND 25 VARIABLES.
NOTE: THE PROCEDURE DB2EXEC USED 0.40 SECONDS AND 1256K.

EXPLAIN PLAN SET QUERYNO=1;
WHERE C.MEMB_ID = S.MEMB_ID
AND C.CLM_ADJU_DATE=S.CLM_ADJU_DATE
AND C.SYS_TIME=S.SYS_TIME
AND S.SERV_PROV_ID = P.PROV_ID
AND S.SERV_PROV_SUFX_ID = T.PROV_SUFX_ID
AND P.PROV_SPEC_CODE = T.PROV_SPEC_CODE
AND C.FST_6_CLKM_ID = '008363';

NOTE: FIELDS REQUIRED FOR UPDATE WERE NOT ENTERED OR IN ERROR
NOTE: TABLE= OPTION REQUIRED FOR PROCESSING WAS NOT ENTERED
NOTE: THE PROCEDURE DB2UTL USED 1.02 SECONDS AND 1212K.

8 * RUN DB2UTL TO RUN THE EXPLAIN FOR SELECT STATEMENT;
10 PROC DB2UTL SSID=DSP;
12 SQL
13 EXPLAIN PLAN SET QUERYNO=1
WHERE C.MEMB_ID = S.MEMB_ID
AND C.CLM_ADJU_DATE=S.CLM_ADJU_DATE
AND C.SYS_TIME=S.SYS_TIME
AND S.SERV_PROV_ID = P.PROV_ID
AND S.SERV_PROV_SUFX_ID = T.PROV_SUFX_ID
AND P.PROV_SPEC_CODE = T.PROV_SPEC_CODE
AND C.FST_6_CLKM_ID = '008363';

NOTE: FIELDS REQUIRED FOR UPDATE WERE NOT ENTERED OR IN ERROR
NOTE: TABLE= OPTION REQUIRED FOR PROCESSING WAS NOT ENTERED
NOTE: THE PROCEDURE DB2UTL USED 1.02 SECONDS AND 1212K.

The PLANNO variable indicates that the CLAIM table is accessed first and ACCESSTY indicates that it does so by a tablespace scan. This table has approximately 30 million rows so we know that this SELECT statement cannot be executed in its current form. This program would run for days! The next example, 3, shows the correct way to code this SELECT statement in order to use an index for the CLAIM table.

4.3 SQL EXPLAIN Statement Example 3
As shown in example 2, data is required for a particular claim id. The problem with using the claim id in the WHERE clause of the SELECT statement is that it is not an index. The DB2 Optimizer is more likely to access a table using an index if that index is used as selection criteria in the WHERE clause of the SELECT statement. Therefore, in addition to using the claim id in the WHERE clause, the member id can also be used. The SELECT statement is changed to reflect this and the EXPLAIN program is ran again.

1 * RUN DB2UTL TO DELETE CONTENTS OF PREVIOUS EXPLAIN;
3 PROC DB2UTL SSID=DSP;
5 SQL DELETE FROM TSID717.PLAN_TABLE;
6 NOTE: FIELDS REQUIRED FOR UPDATE WERE NOT ENTERED OR IN ERROR
NOTE: TABLE= OPTION REQUIRED FOR PROCESSING WAS NOT ENTERED
NOTE: THE PROCEDURE DB2UTL USED 1.02 SECONDS AND 1212K.

8 * RUN DB2UTL TO RUN THE EXPLAIN FOR SELECT STATEMENT;
10 PROC DB2UTL SSID=DSP;
12 SQL
13 EXPLAIN PLAN SET QUERYNO=1
WHERE C.MEMB_ID = S.MEMB_ID
AND C.CLM_ADJU_DATE=S.CLM_ADJU_DATE
AND C.SYS_TIME=S.SYS_TIME
AND S.SERV_PROV_ID = P.PROV_ID
AND S.SERV_PROV_SUFX_ID = T.PROV_SUFX_ID
AND P.PROV_SPEC_CODE = T.PROV_SPEC_CODE
AND C.FST_6_CLKM_ID = '008363';

NOTE: FIELDS REQUIRED FOR UPDATE WERE NOT ENTERED OR IN ERROR
NOTE: TABLE= OPTION REQUIRED FOR PROCESSING WAS NOT ENTERED
NOTE: THE PROCEDURE DB2UTL USED 1.02 SECONDS AND 1212K.
5.0 Conclusion

Computer performance is a major consideration when extracting data stored in DB2 tables, especially when the tables are large. To save computer resources and user time, the SQL EXPLAIN statement should be executed and the contents of the PLAN TABLE reviewed, prior to running a program against a DB2 database.

For more information consult:


I can be contacted at:

Medicare Medical Review
8320 Craig St., Suite 120
Indianapolis, IN 46250-6128
Phone: (317) 841-4660