ABSTRACT

This paper describes the combined usage of ORACLE and SAS to present adverse experience (AE) data for large clinical trials. Large-scale clinical trials are characterized by the large volume of data which is collected from multiple centers with different study designs. The majority of the database usually consists of information related to safety. Among all the safety data collected, adverse experience is some of the most important information corresponding to the safety of a patient. Adverse experience data collected on case report forms includes at least the following: date/time of the occurrence, severity, course, duration, relationship to the study drug and the description of a particular adverse experience. Adverse experiences can be presented in many different ways. Most commonly, they are grouped by body systems and the number of patients reporting these AEs are tabulated side by side for each treatment group. However, adverse experience data has to be first associated with treatment information. This can be easily done with SQL, which is fully supported by ORACLE. Non-standard queries about adverse experience data can then be answered with simple SQL statements. Furthermore, SQL can also be used to generate data sets containing all the relevant information to produce AE summaries. SAS PROC TABULATE is then used to display the data sets in an organized, presentable form. Combining usage of SQL and SAS enables us to generate complicated and non-standard reports with a few lines of SQL and SAS statements. Integration of ORACLE and SAS provides us a means for the preparation of many different types of adverse experience tables, planned or unplanned, with limited resources in timely fashion.

ORACLE DBMS

ORACLE is a relational database management system which provides full support for the relational data language SQL, a de facto standard, and can be run on a wide variety of CPUs including IBM and VAX machines. The relational DBMS provides a logical view of data using tables with rows and columns, with the rows corresponding to records, and the columns representing fields with the records (two-dimensional tables). Users request information by telling the computer what they want, not where it is located physically and how it should be obtained. This feature makes the ORACLE database readily accessible to the end user who may have little or no knowledge of computer programming. The very same feature also may make application programming more productive, since programmers can concentrate on the logical view of the data rather than concerning themselves with the details of physical data representation.

ORACLE supports the minimum set of six operations required for completeness as a relational DBMS: SELECTION, PROJECTIONS, JOIN, DIFFERENCE, DUPLICATE ELIMINATION, and UNION. ORACLE features utilities to manage the execution (SQL-FORMS), and to interface with third-generation languages such as FORTRAN, C, PASCAL, COBOL (PRO+SQL). It also supports facilities such as a data dictionary, data security, concurrent access control and data recovery.

SQL*Plus or SQL*Calc can be used to generate simple reports. For complicated reports with variable numbers of columns or/and rows, PRO+SQL may be used.

CLINICAL TRIAL DATABASE

Clinical Trial is defined by S. J. Pocock as:

"Any form of planned experiment which involves patients is designed to elucidate the most appropriate treatments of future patients with a given medical condition."

There are three main phases of experimentation in a clinical trial before any Investigational New Drug (IND) can be filed as a New Drug Application (NDA). Phase I experiments are usually performed on healthy male volunteers between the ages of 18 and 45 to study the toleration of the individuals against the new drug. Phase II trials are usually dose range studies. Once a test drug is shown to be somewhat effective, it will undergo Phase III experiments, which are designed to study not only the efficacy but, just as important, the safety of the drug in a wide range of patient population.

Each clinical study is conducted under a protocol which is designed by the Pfizer clinician with input from statisticians and other staff. Each new drug project might require anywhere from 10 to 80 different protocols. Each protocol might involve hundreds of patients in the study, and the design of the study varies from single dose, open, non-comparative, double blind parallel, crossover or randomized withdrawal.

At Pfizer, the bulk of the data from different phases of studies is received from investigators on signed Case Report Forms (CRFs). Data on CRFs is entered interactively by trained data entry personnel. Double data entry by a different person is usually done for key data such as numeric data. Before data is actually entered into the database, manual checks by Clinical Research Associates (CRAs) or Clinical Data Analysts (CDAs) on CRFs are performed for accuracy (does the data comply with the protocol?) and completeness (is there any missing data?). Data which is received on disks or tapes can be entered through the batch process.

Data is initially maintained in UPDATE tables, temporary holding areas in the database, to await validation. Once verified, the data is then transferred or merged into the DATA tables at the appropriate time. Further suspect data will remain in the temporary areas until further clarification. Any changes to the DATA tables will result in a copy of the original record, datetime stamp, and user identification being placed into the AUDIT table (audit trail entry).

Simple encoding is usually performed at the level of data entry. Further encoding is performed automatically by matching the codes in the database with those of codelists or data dictionaries. Any non-matching text or codes are
stored in separate files for review. Data checking can be performed periodically via the Pfizer Edit Check System (PECS). The result is a "clean" database which can be used directly in ad hoc queries with simple SQL statements and/or be processed to generate listings, reports, or statistical analysis for review by clinicians, CDA's or for NDA submission purpose.

ADVERSE EXPERIENCE DATA

Among all the clinical data collected, safety data of the patients is one of the most complicated and important pieces of information. Adverse experience (AE) data and laboratory data make up an integral part of the safety data. AE data is usually collected on case report forms, and includes at least the following: patient identification, date/time of the occurrence, severity, causality (relationship of the side effect to the study medication), onset and/or stopped date of the side effects, investigator's description of AE and course of the side effects (e.g., dose reduced or dose stopped, etc.). Also recorded is the information on treatment, demographic characteristics, concomitant medications, lab abnormalities and randomization sequences, etc.

There are many ways to present AE data. One of the most common and straightforward methods is to present side effects in terms of number and percentage of patients with AE during the study (Figure 1). Since a patient may report the same side effect more than once during the study, to produce such a table, a patient can be counted only once for each side effect. SQL can achieve this with very simple statements (Figure 2). With the same type of "counting" capability, SQL can also easily derive the number of patients in the study, number of patients with side effects, or number of patients withdrawn due to side effect. Other types of ad hoc queries such as numbers of male patients with side effects, or numbers of patients with headache can also be answered with simple SQL statements (Figure 3).

Indeed, almost all the AE tabulations are derived from the same basic idea, i.e., counting the number of patients under different conditions. For example, similar tables can be generated, but only with counts of treatment-related side effects (Figure 4), or total counts of patients in terms of worst outcome, or maximum severity during the study. Similarly, tables with older patients vs. young, male vs. female, or patients with renal disease or diabetics vs. those without such disease can also be generated. With its flexibility and powerful "counting," "sorting" and "joining" capability, SQL is an excellent tool to generate data sets which contain all the necessary information to be used in procedural language such as SAS or Fortran to produce the final report or statistical analysis.

Since the ultimate purpose of collecting AE data is to assess the safety of the investigational drug, treatment information (treatment groups, duration of therapy, actual drug/dose intake) has to be associated with AE data first. The adverse records have to be "reduced" in several steps before the counting of adverse experience can proceed. First, each adverse event will be cross-referenced to an appropriate term in the WHO (World Health Organization) dictionary; thus, adverse events with different investigator's entries may be treated as the same side effect. Also, causality, severity and course of AE data will be designated as the worst recorded during the treatment period, i.e., if the same side effect was reported several times over the same period, then this side effect would be collapsed into one single record with the worst causality, severity and course. The so-called "data reduction" of AE records can be done easily with simple SQL statement.

GENERATION OF STANDARD AE REPORTS BY END USERS USING ADRS

Over the past year, Pfizer has developed a new version of Adverse Drug Reaction System (ADRS) based on ORACLE database management system (Figure 5). ADRS was specifically designed to generate standard side effects reports which include summary of side effects counts, the profile of severity of side effects, and displays of patient counts and their percentage.

Several formal or informal interviews with end-users were conducted before the actual design began. The content of the new ADRS is primarily based on initial user's requests and, most importantly, feedback during the use of the earlier version of ADRS. Prior to the system implementation, design documentation including the algorithm and priority of the implementation was widely distributed within the Clinical Research Department. The comments received also contribute heavily to the contents of the system.

Primarily due to users' requirements, and partly due to logical structures of the clinical database, ADRS was mainly designed to generate reports with input data from individual protocols. AE reports with data pooling from multiple protocols can also be generated.

For each protocol, more than fifty different AE reports with different combinations of report parameters can be generated. Report parameters include the following information: drug project number, Protocol number, report type, study type, study period, relationship to the treatment, etc. (Figure 6). A simple SQL "where clause" which functions as the qualifier for input tables, may also be included (Figure 7). Report parameters for each report were entered and maintained by end-users through SQL*Forms. ADRS provides default values for most of the report parameters, but end-users can overwite those values. Also provided is a built-in checking mechanism. If erroneous values are accidentally entered, ADRS will send the warning message and will not move the cursor to the next parameter field. Help messages are available at the user's request.

Once this information is entered, it is stored permanently in ORACLE tables; thus, the same reports with the same or different input data can be regenerated easily. Reports can be generated interactively or through batch process. SQL statements of "joining" and "counting" were used extensively within ADRS. SQL was first used to "join" the AE information with treatment information, then to "reduce" those records into a normalized form. Finally, the "counting" capability of SQL was applied to obtain the counts needed to produce the final output.

GENERATION OF OVERALL SUMMARIES

To file a new investigational drug for NDA, not only study summaries from individual protocols have to be submitted. Those summaries with data pooled from multiple protocols must be included in the submission as well. Data in the Pfizer clinical database is stored in such a way that same information from different protocols is stored with the same data item name. With standards such as the one above, being enforced, pooling data across
multiple protocols still presents a number of problems. Each protocol is usually defined by a study design with data collected in a different manner. With different study designs, report parameters such as washout period and extend duration of therapy may be different. Also, patients in some of the protocols may have illness before the study begins; those symptoms may be suppressed from reporting even if they were recorded on CRFs. Thus, data from each protocol has to be uniquely *rectangular* and *processed* in a manner so that a *"flat*" or *"rectangular* file with a standard fixed format can be generated before it is merged with data from other protocols.

Recently, more than 50 different types of overall AE reports were generated for a new Pfizer investigational drug which contains more than 3,500 patients with chronic cardiovascular disease, with an average of 30 protocols being combined in each report. Most of the overall summaries are different, either in their contents or formats, from those of reports for single protocols. The project to generate all the overall summaries has to be fully implemented and operational in less than 3 months.

One of the most common methods to complete a computing project in time is to add manpower at the early stage of the project. But to add new team members may not only complicate the coordination and control of the project, also precious time may have to be allocated to train new personnel to familiarize themselves with the problems, which is the most difficult part of being an application programmer. In addition, new programmers may also have to learn programming tools. In order to save time, a system was designed in such a way that new programmers were insulated from the details of the problems; instead they concentrated on producing specific reports with proper data sets. The proper programming tools which would be easy to use also had to be chosen. The easier the programming tools, the faster the new programmer can eventually contribute to the project.

Even though SAS was originally developed for statistical needs, it does provide extraordinary data manipulation capabilities. A few lines of SAS statements can accomplish tasks which may require pages of code written in third-generation languages such as Fortran, and SAS can be easily self-taught. The report-generation facilities in SAS and PROC TABULATE, in particular, have proved to be exceptionally powerful and flexible. Most of the AE summaries with fixed formats were requested before the project began, but some unplanned summaries were requested after the outputs were being reviewed, and some tables which were requested early might need modifications of format. The fact that SAS codes can be modified easily ultimately made us select the SAS system as the programming tool for formatting.

"Rectangular" files which are the result of a direct dump either from the Pfizer Clinical database tables or *"JOINED"* ORACLE tables, can be read directly into SAS data sets. With the input of a preprocessed SAS data set, complicated reports can be generated with a few lines of PROC TABULATE statements (Figure 8) and with a minimum amount of programming effort. Results can be further verified with a simple ORACLE *SELECT* operation.

Batch processes were set up to generate *"rectangular* files. Report parameters needed were stored in separate files so that they could be accessed directly by DCL (DEC Command Language) files to generate the desired output files. *"Rectangular* files were produced on a per-protocol basis. One of the apparent advantages is that if corrections of data are made on one of the protocols, the corresponding *"rectangular* file could be regenerated immediately. These files are stored in a *"library* account used by several programmers as input to produce various types of reports. It was up to each programmer to extract data from files of different combinations of protocols to generate overall summaries. Also, centralized codewriters which are updated regularly are accessed by all the programs. With proper system design, easy-to-use programming tools such as SQL and SAS, and teamwork, we were able to finish the project of such magnitude in time (Figure 9).

CONCLUSION

SQL provides a powerful tool in creating organized and ready-to-be-processed files from data with complicated structure such as adverse experience data from a clinical trial. SAS, with its powerful and flexible procedures like PROC TABULATE, produces all the adverse experience tabulations with very little programming. Since all the *"coding* was done with high-level languages, the programs which produce the output are *"self-documented* and need very little documentation, and are therefore very easy to maintain. Verifying the final results is usually straightforward since most of the programs are very short in length as well as simple in logic. The integrated use of ORACLE and SAS enabled Pfizer to generate an important part of an NDA submission--summaries of adverse experiences, in a timely fashion with minimum manpower.

ACKNOWLEDGEMENTS

The authors would like to thank the following individuals for the invaluable assistance during the preparation of AE reports: Nancy Dirilm, Rick Pezzullo, Althea Sergeant, Terrie Wood, Lonni Zubkoff. We also wish to acknowledge Dr. Joan P. Leader who reviewed the article. Finally, we wish to thank Jane Kenny who skillfully typed the final copy of the paper.

REFERENCES

2. Chang, C., *"ADRS Design Specifications"*, Pfizer Internal Documentation. 1986
5. Pocock, Stuart J. *"Clinical Trials, A Practical Approach*", 1983, John Wiley & Sons Ltd.

Glossary:

ADRs Adverse Drug Reaction System
AE Adverse Experience
CDA Clinical Data Analyst
CRA Clinical Research Associate
CRF Case Report Form
FOR further information or comments, please contact either author at:

Clinical Research Department
Pfizer Central Research
Eastern Point Road
Groton, CT 06340

Key Words: ADRS; Adverse experience; Clinical trials; DBMS; NDA; ORACLE; PROC TABULATE; Side effects; SQL; Study summaries.

<table>
<thead>
<tr>
<th>TREATMENT-RELATED SIDE EFFECTS, DRUG XXXX, PROTOCOL YYYY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.25 MG XXXX</td>
</tr>
<tr>
<td>---------------</td>
</tr>
<tr>
<td>NO. (%)</td>
</tr>
</tbody>
</table>

**NUMBER OF PATIENTS:**

| Evaluable | 41 (17.1%) | 42 (11.9%) | 44 (9.1%) | 43 (9.3%) | 40 (10.0%) |

| Evaluable with side effects | 7 (17.1%) | 5 (11.9%) | 4 (9.1%) | 4 (9.3%) | 4 (10.0%) |

| Evaluable with withdrawn side effects | 0 (0.0%) | 1 (2.4%) | 0 (0.0%) | 1 (2.3%) | 0 (0.0%) |

**CARDIOVASCULAR**

- Palpitations: 0 (0.0%) 0 (0.0%) 1 (2.3%) 1 (2.3%) 0 (0.0%)
- Edema Dependent: 0 (0.0%) 0 (0.0%) 0 (0.0%) 2 (4.7%) 1 (2.5%)

**SKIN AND APPENDAGES**

- Pruritus: 0 (0.0%) 1 (2.4%) 0 (0.0%) 0 (0.0%) 0 (0.0%)

**CENTRAL & PERIPHERAL NERVOUS SYSTEM**

- Dizziness: 2 (4.9%) 1 (2.4%) 0 (0.0%) 1 (2.3%) 1 (2.5%)
- Headache: 3 (7.3%) 4 (9.5%) 1 (2.3%) 2 (4.7%) 3 (7.5%)

**AUTONOMIC NERVOUS SYSTEM**

- Sweating Increased: 1 (2.4%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%)

**PSYCHIATRIC**

- Somnolence: 2 (4.9%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%)
- Constipation: 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%)

**GASTRO-INTESTINAL**

- Nausea: 1 (2.4%) 2 (4.8%) 1 (2.3%) 0 (0.0%) 0 (0.0%)

**URINARY SYSTEM**

- Micturition Frequency: 1 (2.4%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%)

**GENERAL**

- Fatigue: 1 (2.4%) 0 (0.0%) 0 (0.0%) 0 (0.0%) 0 (0.0%)
- Pain: 0 (0.0%) 1 (2.4%) 1 (2.3%) 0 (0.0%) 0 (0.0%)

**Figure 1**
Figure 5

PROC TABULATE:
CLASS DRGGROUP BCODE AECODEBI;
VAR COUNT PERCENT;
LABEL BCODE = 'Organ System';
   PERCENT = 'N';
   COUNT = 'N';
DRGGROUP = 'TESTDRUG REGIMEN';
   AECODEBI = 'SIDE EFFECT';
FORMBT BCODE BCODEF. DRGGROUP GROUPF.:
   AECODEBI CODEF.:
   TABLE BCODE BCODEF. ALL,
   DRGGROUP * (COUNT F=6. PERCENT F=8.)
   KEYLABEL SUM =
      ALL = 'TOTAL';

Figure 7

Figure 8