Survey Methodology, December 1988
Vol. 14, No. 2, pp. 327–330
Statistics Canada

A Brief Note on SQL

DAVID N. EMERY[1]

ABSTRACT

This note portrays SQL, highlighting its strengths and weaknesses.

KEY WORDS: Relational database management system; Database query language.

1. INTRODUCTION

A great deal of media attention has been focused on relational database management systems and SQL (pronounced see-quel), the most popular of the associated database query languages. To a large extent, SQL has been cast in the role of panacea for all the ills associated with data management. Unfortunately, this leads to a great deal of misconception on the part of potential users of SQL. These people are then sometimes disappointed with SQL when they eventually get a chance to use it.

The intent of this note is to clear up some of this misconception by providing a realistic portrayal of SQL, highlighting its inherent strengths and weaknesses. No attempt will be made to elaborate the advantages of the relational data model itself. These advantages have been adequately documented elsewhere (Date 1985).

2. SQL - WHAT IS IT?

The interaction which takes place between a user (whether systems developer or end user) and a database management system can be broadly categorized according to the function taking place:

- data definition;
- data control (*i.e.* authorization and control of data integrity);
- data retrieval; and,
- data modification (*i.e.* insert, update, and delete).

A database management system must provide interfaces for carrying out each of these functions. Depending on the particular system, these interfaces take the form of utilities, query languages, and/or subroutine libraries for programming languages.

SQL addresses these four functions in a single well-defined, rigidly structured language. SQL is the interface used to communicate, to the database management system, how relations (*i.e.* logical files or tables) are to be subdivided and/or combined to create new relations.

The key to understanding SQL's capabilities is an appreciation of the fact that SQL addresses exactly these four roles - no more and no less. Any other functionality must be supplied by the application which initiates the SQL statement.

Consider the following example. The table, DWELLING, contains information about dwellings such as number of occupants, type of dwelling, where it is located, type of heating,

[1] David N. Emery, Statistics Canada, Research and General Systems Subdivision, Room 2405, Main Building, Tunney's Pasture, Ottawa, Ontario K1A 0T6.

and age of dwelling. In order to impute the type of dwelling, one might want to obtain a set of potential donor dwellings which are in the same geographic area, are the same age, and use the same heating method. The following SQL statement could be issued to obtain a donor set:

```
SELECT DWELLING_ID, TYPE_OF_ DWELLING                    (Query 1)
  FROM DWELLING
  WHERE HEATING_ TYPE = 'GAS' AND
    AGE = 20 AND
    LOCATION_ CODE = 'XYZ';
```

SQL does not provide a mechanism for manipulating the set of retrieved donor records. Selecting the n'th record, every second record, or a random record are all beyond the capability of SQL. Similarly, SQL has no mechanism for manipulating a table to affect its appearance on a terminal or printer. These are capabilities one would rightfully demand of a programming language, and hence the term database query language. Calling SQL a fourth generation language (4GL), then comparing it to products which incorporate only the data retrieval and data modification functions into a programming language, only adds to the confusion. It is really an apples and oranges comparison since both are 4GLs, but of very different flavours.

Given this very focused functionality, the obvious question then has to be — why all the fuss about SQL?

3. SQL — ITS BENEFITS

3.1 Implementation Transparency

A SQL query indicates nothing about how the data is actually organized and stored on the database. The query states what is to be retrieved, modified, or stored; the database management system determines the best way to do it. Issues such as:

- which data columns are indexed (a performance improvement feature);
- whether the table/column is actually stored or merely an execution time combination of other tables; and,
- the data's internal representation (*i.e.* floating point, packed decimal, binary)

have no bearing whatsoever on a SQL statement's syntax. Consequently, the user is immune to changes in the database's organization and structure. Changes to the underlying structure of the database can be made at will without changing the query. A query is immediately able to take advantage of improvements in the database structure or optimization algorithms.

Similarly, when formulating a SQL query the user does not specify the order in which processing is to take place to satisfy the query. That is the responsibility of the query processing software's optimization algorithms. This software evaluates the query against the current structure and organization of the database to determine the most efficient way of satisfying it.

3.2 Non-proprietary, Internationally Accepted Standard

Both the International Standards Organization (ISO) and the American National Standards Institute (ANSI) have recently adopted a common standard for SQL (ISO 1987). The existence of this standard, with a commitment to it by a number of relational database management system vendors, gives software developers access to a much broader market without significantly extra development effort. By building their applications on top of standard SQL, they have removed their reliance on a particular database management system. As a result, the creation

of software tools, built upon an interface to this standard version of SQL, has become a major growth industry. For example, natural language interfaces, fourth generation programming languages, data dictionary software, data entry/validation packages, and spreadsheet software, all layered on top of ANSI/ISO SQL, are beginning to appear on the market.

The active interest in SQL has also had a very positive impact on the SQL standard itself; it is continuing to evolve. The most recent draft revision to the ISO Standard for SQL incorporates the specification of referential integrity constraints into SQL's data definition statements. The significance of this extension to SQL is best illustrated by a further elaboration of the DWELLING example. Assume that the database also has a table PERSONS which contains detailed information about individuals including a dwelling code which indicates the dwelling where they currently reside. One might define a integrity constraint stipulating that each person must be associated with exactly one dwelling. Consequently, it would be an error to delete a DWELLING record which still had any PERSONS records referencing it, or to add a PERSONS record which referenced a nonexistent DWELLING record. Currently, logic to detect and prevent these inconsistencies must be inserted into each application program capable of deleting a DWELLING record. With the incorporation of referential integrity specifications into SQL, this program logic will no longer be required. The DBMS software assumes responsibility for detecting and terminating any attempt to remove a DWELLING record which still has associated PERSONS records.

3.3 Ease of Extension

One of the major differences between the various vendors' versions of SQL is the number and variety of supported functions. This is to a large extent due to the ease with which extra functionality can be incorporated into SQL, without change to its overall structure. For example, the SQL standard documents the grouping functions of average (AVG), maximum (MAX), minimum(MIN), enumeration (COUNT) and aggregation (SUM) for unweighted data. Referring again to the earlier DWELLING example, one could generate various summary statistics about number of occupants, broken down by geographic location:

```
SELECT AVG (NO_OF_OCCUPANTS), MAX (NO_OF_OCCUPANTS),(Query 2)
  MIN (NO_OF_OCCUPANTS), SUM (NO_OF_OCCUPANTS),
    COUNT (NO_OF_OCCUPANTS)
FROM DWELLING
GROUP BY LOCATION_CODE;
```

Some Vendors have augmented these functions with others such as variance (VARIANCE) and standard deviation (STDDEV). With these extra functions the identification of outliers, more than one standard deviation from the mean, is a very straightforward exercise:

```
SELECT DWELLING_ID FROM DWELLING                              (Query 3)
  WHERE NO_OF_OCCUPANTS <
    (SELECT AVG (NO_OF_OCCUPANTS) - STDDEV (NO_OF_OCCUPANTS)
      FROM DWELLING)
  OR
    NO_OF_OCCUPANTS >
    (SELECT AVG (NO_OF_OCCUPANTS) + STDDEV (NO_OF_OCCUPANTS)
        FROM DWELLING);
```

3.4 Single Interface to the Database

When interrogating a database from within a host language program such as PL/1, FORTRAN, or C, one also uses SQL statements. These statements are virtually identical to

those used when interrogating the database interactively via a SQL statement processor. The only difference lies in the fact that the host language interface requires an additional INTO clause to indicate the program variables receiving the results of the query.

By using an identical interface to a host programming language, one is able to separate the program development and debugging exercise into two distinct activities:

- testing of the database retrieval storage statements (*i.e.* the SQL statements themselves), and
- testing of the program code which manipulates the data.

The first of these activities can be carried out using a SQL command interpreter even before the host language program has been written. The optimal SQL statements can then be moved directly into the host program where the testing effort can be focused on the logic associated with manipulating the data.

Since the SQL statements embedded in the host language are interpreted at execution time, any changes made to the database organization or structure are immediately reflected in the program.

3.5 Suitability for Distributed Databases/Database Machines

One of the hottest topics in database management systems technology today is distributed databases. In a distributed database environment, the data is spread across a number of different databases (often on physically separate machines). It is the DBMS software's responsibility to intercept a user's query, translate it into appropriate queries to the various constituent databases, and assemble the results of these queries for presentation.

As discussed earlier, a SQL statement is devoid of constructs associated with describing how or where the data is stored on the database. Consequently, in a distributed database environment where SQL is used as the database query language, data can be moved between machines with no change whatsoever to existing applications. SQL is therefore becoming quite popular with the developers of distributed database management systems.

For similar reasons, SQL is gaining popularity as a query language for database machines. These machines take advantage of relational (*i.e.* tabular) data structures' inherent regularity to partition them across a number of parallel processors. These processors have instruction sets specifically designed to perform relational operations. The lack of representational detail in SQL queries completely insulates users from an awareness of what these machines are doing behind the scene.

4. SUMMARY

There is no question that SQL has quickly become the pre-eminent database query language. The database management system which does not feature a SQL interface will soon be the exception. An interesting anomaly will however emerge. The user will, over time, see less and less of SQL. Rather than trying to make SQL itself a user-friendly language, effort will be focused on the devlopment of application specific tools which provide the user with an interface tailored to the task at hand. SQL will be the common interface between these tools and the various databases.

REFERENCES

DATE, C.J. (1985). *An Introduction to Database Systems.* Don Mills: Addison-Wesley.

INTERNATIONAL STANDARDS ORGANIZATION (1987). Database Language SQL. International Standards Organization 9075.