A History and Evaluation of System R

The history and evaluation of an experimental relational database system, System R, which demonstrates that it can incorporate the high performance and complete function required for everyday production use has been discussed in 3 phases:

Phase Zero- An Initial Prototype: The base was XRM, a single-user access method without locking or recovery capabilities. An interpreter program was written in PL/I to execute statements in the high-level SQL language on top of XRM. Issues relating to concurrency and recovery were excluded in this phase. One of the basic design decisions was that the system catalog should be stored as a set of regular relations in the database itself. The optimizer algorithms were designed to minimize the number of tuples fetched from the database in processing a query. This phase demonstrated the usability of SQL language, the feasibility of creating new tables and inversions “on the fly” and relying on an automatic optimizer for access path selection, and the convenience of storing the system catalog in the database. It also gave an insight into better measures to determine the costs for optimizing. The “join” formulation of SQL was considered. It was also realized that simple interactions were better and to minimize the path length for simple SQL statements in further models.

Phase One- Construction of Multi-user Prototype: The prototype consists of two modules, Research Storage System (RSS) – an access method where all locking and logging functions were isolated – and Relational Data System (RDS) – an optimizing SQL processor that has all authorization and access path selections.

The high level SQL language statements were complied into small, efficient machine-language routines which are packaged into an ‘access module’ that is invoked when a program is executed to perform all interactions with the database by means of calls to the RSS. Data values were stored in the individual records of the database and could be fetched by a single I/O. RSS provided ‘indexes’ and ‘links’ as access aids. The optimizer was designed to minimize the weighted sum of the predicted number of I/Os and RSS calls in processing an SQL statement two kinds of join methods were implemented. View definitions were based on SQL parse trees and authorization subsystems were based on SQL statements GRANT and REVOKE. System R uses of “dual logs” and “shadow pages” helps it in media, system and transaction recovery. The locking subsystem involves a hierarchy of locks with several different sizes of lockable units.

Phase Two – Evaluation: The general user comments were satisfactory. SQL user interface was found to be successful in achieving its goals of simplicity, power and data independence. The compilation approach was also successful and they were able to generate a machine-language routine to execute any SQL statement of arbitrary complexity by selecting code fragments from the library. The optimizer was able to correctly order access paths but it was not structured to consider multiple uses of an index within a single query. Views and authorization were found to be powerful, flexible and convenient. The combined “shadow page” and log mechanism was quite successful though it limited the clustering ability, required directories to be maintained for locating “old” and “new” versions of each page which also consumed certain CPU time thus affecting the performance on the whole. The locking system was implemented by providing the user with three levels of isolation. The ‘Convoy problem’ arose during the locking phase and it was addressed.

The authors thus conclude by saying that System R demonstrated the feasibility of applying a relational database system to a real production environment in which many concurrent users are performing ad hoc queries and repetitive transactions. They feel that a relational system is more likely is able to adapt to a broad spectrum of unanticipated applications (with adequate performance) when compared to navigational system.