Abstracts

Listed are abstracts from recent papers by IBM authors. Inquiries should be directed to the publications cited.

Empirical data reference behavior in data base systems, J. Rodríguez-Rosell, (RES San Jose, CA), *Computer* **9,** No. 11, 9–13 (November 1976). During the past several years, a considerable amount of effort has gone into the measurement, analysis, and modeling of program behavior. Most of the work either assumes the existence of a reference string or attempts to produce a model for the process by which these reference strings are generated. The author extends this methodology to the study of data base systems. Data base reference strings are found to exhibit strong sequentiality in addition to weak locality.

Experiments in text file compression, F. Rubin (SPD Poughkeepsie, NY), *Communications of the ACM* **19,** No. 11, 617–623 (November 1976). A system for the compression of data files, consisting of an encoder, an analysis program, and a decoder, is presented. Data files are viewed as strings of characters, making the method general, applying equally well to English, to PL/I, or to digital data. The author finds a high degree of text compression can be obtained if fairly high computation time and large storage are available.

Incremental program testing in a very high-level language, B. M. Leavenworth (RES Yorktown Heights, NY), *ACM '76, Proceedings of the Annual Conference*, 499–503. A testing and debugging methodology is presented which exploits the following properties of very high-level languages: functionality, single assignment property, locality of reference, and aggregate operations. The approach is based on incremental construction of a program with testing and debugging in parallel using a graphic display and light pen. It is shown how these properties allow execution of arbitrarily small phrases of the program to obtain aggregate values. The emphasis on data flow rather than control flow permits causal errors to be traced by bottom-up or top-down scans of the program tree. Examples of the methodology are given using the Business Definition Language.

An introduction to proving the correctness of programs, S. L. Hantler and J. C. King (RES Yorktown Hts., NY), *ACM Computing Surveys* **8,** No. 3, 331–353 (September 1976). Interest in verifying that computer programs behave as they were intended to behave has existed since the advent of modern electronic computers. This paper explains, in an introductory fashion, the method of specifying the correct behavior of a program by the use of input/output assertions and describes one method for showing that the program is correct with respect to those assertions. An initial assertion characterizes conditions expected to be true upon entry to the program, and a final assertion characterizes conditions to be true upon exit from the program. When a program contains no branches, a technique known as symbolic execution can be used to show that the truth of the initial assertion upon entry guarantees the truth of the final assertion upon exit. More generally, for a program with branches, one can define a symbolic execution tree.

Modeling and performance evaluation of physical data base structures, S. B. Yao (RES San Jose, CA), *ACM '76, Proceedings of the Annual Conference*, 303–309. A generalized file organization model and performance evaluation system is developed for estimating the performance of physical data base structures. Performance evaluation results based on the cost functions of the model and system are presented. These performance measures are compared with the results of previous simulation models for indexed sequential, multi-list, and inverted file structures. The analytic approach makes the costs of evaluation very low. Consequently, many evaluations may be performed interactively with the file designer when searching for the most suitable structure for a given application.

The notions of consistency and predicate locks in a database system, K. P. Eswaran, J. N. Gray, R. A. Lorie, and I. L. Traiger (RES San Jose, CA), *Communications of the ACM* **19,** No. 11, 624–633 (November 1976). In data base systems, users must access shared data under the assumption that the data satisfies certain consistency constraints. This paper defines concepts and shows that consistency requires that a transaction cannot request new locks after releasing a lock. From this it is argued that a transaction needs to lock a logical rather than a physical subset of the data base. An implementation of predicate locks which satisfies the consistency condition is suggested.

On user criteria for data model evaluation, W. C. McGee (GPD Palo Alto, CA), *ACM Transactions on Database Systems* **1,** No. 4, 370–387 (December 1976). A data model is the class of logical data structures that a computer system or language makes available to the user for the purpose of formulating data processing applications. The diversity of computer systems and languages has resulted in a corresponding diversity of data models and has created a problem for the user in selecting a data model which is appropriate to a given application. An evaluation procedure is needed which will allow the user to evaluate alternative models in the context of a specific set of applications. This paper takes a first step toward such a procedure by identifying the attributes of a data model which can be used as criteria for evaluating the model. The use of the criteria is illustrated by applying them to three specific models: an *n*-ary relational model, a hierarchic model, and a network model.

Optimal reorganization of distributed space disk files, K. Maruyama and S. E. Smith (RES Yorktown Hts., NY), *Communications of the ACM* **19,** No. 11, 634–642 (November 1976). In most data base organizations, the cost of access will increase due to structural changes caused by insertions and updates. These access costs can be reduced by reorganizing the data base. Therefore, the user must establish the proper trade-off between performance, storage costs, and reorganization costs. This paper considers the optimum points at which to reorganize a data base. A disk file organization which allows for distributed free space is described. A cost function describing the excess costs due to physical disorganization is defined, and this function is minimized to obtain the optimum reorganization points. Numerical examples are given, based on the characteristics of existing disk storage devices.

Replacement algorithms for storage management in relational data bases, R. G. Casey (RES San Jose, CA) and I. M. Osman (Durham University, Durham, UK), *The Computer Journal* **19,** No. 4, 306–313 (November 1976). This paper treats a generalized storage management problem in which the pages have varying sizes and the cost of a page fault is a function of the particular page reference. In such an environment, the conventional page replacement algorithms are found to perform inadequately, so new ones are proposed. One practical environment in which the general problem may arise is a relational data base having an implied relations facility, in which some relations are maintained in definition form until queried. The implied reaction is analogous to a page, and the processing time for restructuring a relation from its definition varies from one relation to another. A suitable replacement algorithm is needed to manage and process as the implied relations alternate between the state of definitions and the state of explicit representation in a fixed buffer storage area.

Roster of programming languages for 1974–75, J. E. Sammet (IBM Cambridge, MA), *Communications of the ACM* **19,** No. 12, 655–669 (December 1976). This roster contains a list of 167 currently existing higher-level languages which have been developed or reported in the United States; have been implemented on at least one general-purpose computer; and are believed to be in use in the United States by someone other than the developer.

Scheduling as a graph transformation, E. B. Fernández (IBM Scientific Center, Los Angeles, CA) and T. Lang (UCLA, Los Angeles, CA), *IBM Journal of Research and Development* **20,** No. 6, 551–559 (November 1976). The scheduling of a set of tasks, with precedence constraints and known execution times, into a set of identical processors is considered. Optimal scheduling of these tasks implies utilizing a minimum number of processors to satisfy a deadline, or finishing in minimal time using a fixed number of processors. This process can be seen as a transformation of the original graph into another graph, whose precedences do not violate the optimality constraints and has a unique basic schedule. Analysis of this transformation provides insight into the scheduling process and also into the determination of lower bounds on the number of processors and on time for optimal schedules.

Scheduling of unit-length independent tasks with execution constraints, T. Lang (UCLA, Los Angeles, CA) and E. B. Fernández (IBM Scientific Center, Los Angeles, CA), *Information Processing Letters* **4,** No. 4, 95–98 (January 1976). Consider a set of unit-length independent tasks with individual execution intervals which are to be scheduled into a set of identical processors. A scheduling algorithm is presented that requires a minimum number of processors for a given set of execution intervals.

SEQUEL 2: a unified approach to data definition, manipulation, and control, D. D. Chamberlin, M. M. Astrahan, K. P. Eswaran, P. P. Griffiths, R. A. Lorie, J. W. Mehl, P. Reisner, and B. W. Wade (RES San Jose, CA), *IBM Journal of Research and Development* **20,** No. 6, 560–575 (November 1976). SEQUEL 2 is a relational data base language that provides a consistent, English keyword-

oriented set of facilities for query, data definition, data manipulation, and data control. SEQUEL 2 may be used either as a stand-alone interface for nonspecialists in data processing or as a data sublanguage embedded in a host programming language for use by application programmers and data base administrators. This paper describes SEQUEL 2 and the means by which it is coupled to a host language.

Software development, H. D. Mills (FSD Gaithersburg, MD), *ACM '76, Proceedings of the Annual Conference* (supplement), 79–86. Software development has emerged as a critical bottleneck in the human use of automatic data processing. Beginning with ad hoc, heuristic methods of design and implementation of software systems, problems of software maintenance and changes have become unexpectedly large. It is contended that improvement is possible only with more rigor in software design and development methodology. Rigorous software design should survive its implementation and be the basis for further evolution. Software development should be done incrementally, in stages, with continuous user participation and replanning, and with design to cost programming within each stage.

Statistical analysis of non-stationary series of events in a data base system, P. A. W. Lewis (Naval Postgraduate School, Monterey, CA) and G. S. Shedler (RES San Jose, CA), *IBM Journal of Research and Development* **20,** No. 5, 465–482 (September 1976). Central problems in the performance evaluation of computer systems are the description of the behavior of the system and characterization of the workload. One approach to these problems comprises the interactive combination of data-analytic procedures with probability modeling. This paper describes methods for the statistical analysis of processes frequently encountered in computer systems: nonstationary univariate stochastic point processes and sequences of positive random variables. Models developed from these analyses have application to the validation of proposed data base subsystem models.