Jim Gray at IBM

The Transaction Processing Revolution

Bruce G. Lindsay
IBM Almaden Research Center
560 Harry Road
San Jose, CA 95120
bgl@almaden.ibm.com

ABSTRACT

While at the IBM San Jose Research Laboratory, in the 1970's, Dr. Jim Gray defined and developed the fundamental concepts and techniques that underlie on-line transaction processing systems. Jim Gray's pivotal contributions enabled cost efficient, on-line processing to replace paper and batch processing systems. Today, on-line transaction processing powers the record keeping systems that drive today's commerce, services, and government.

1. INTRODUCTION

Dr. Jim Gray worked at the IBM San Jose Research Laboratory from October 1972 until September 1980. During that time he developed and implemented the foundational techniques that underlie and enable on-line transaction processing. The deployment of on-line transaction processing reduces the cost of business transactions by reducing delays and eliminating paper records. Dr. Gray received the 1998 A.M. Turing Award "*For fundamental contributions to database and transaction processing research and technical leadership in system implementation from research prototypes to commercial products. The transaction is the fundamental abstraction underlying database system concurrency and failure recovery. Gray's work [defined] the key transaction properties: atomicity, consistency, isolation and durability, and his locking and recovery work demonstrated how to build ... systems that exhibit these properties.*"

On-line transaction processing provides the critical infrastructure that enables reliable and cost efficient financial, commercial, travel, medical, and governmental operations. Without on-line transactions, life, as we know it, would be quite different (and less pleasant). In order for on-line processing to replace paper based and batch processing record keeping two fundamental problems must be addressed: the electronic records must be reliable in the presence of equipment failures; and interference among multiple, concurrent application programs operating on shared records must be controlled.

In the context of the System R relational database project at IBM Research [Chamb 1981, Blas 1981], Jim Gray developed and refined recovery techniques that ensure the reliability of the records and concurrency control methods to coordinate interactions among simultaneously executing programs accessing and modifying shared sets of records. Not only did Dr. Gray develop new and important technology, he also published and explained his ideas to generations of database users and developers. His "Notes on Database Operating Systems" [Gray 1978] is a classic work familiar to many (if not most) DBMS students and practitioners and the (big !) book [Gray 1983] provides a comprehensive description of "*Transaction Processing: Concepts and Techniques*".

2. TRANSACTION RECOVERY

Jim Gray refined the notion of a Database Transaction. He explained that application initiated data manipulation actions can be classified as "unprotected", "protected", and "real" actions [Gray 1981b]. *Unprotected* actions involve transient and internal state, such as temporary files. *Protected* actions, on the other hand, are grouped into transactions and are reflected in the state of the transaction outcome. The outcome of a transaction must be to either *commit* the effects of its protected actions to the system state, or to *abort* and remove the protected actions' effects from the system state. This means that protected actions must be undone on transaction failure or abort and their effects must be ensured in the case of transaction commit. *Real* actions involve sensors, actuators, and messages outside the DBMS. While real actions cannot be "undone", they can be compensated. For example, if the missile is fired, the compensation could be "debit quantity on hand and send apologies".

In order to achieve durable transaction atomicity (all or nothing for protected actions) in the presence of processor, memory, storage, communication, or environmental failures, multiple copies of the stored data

must be maintained and a record of the protected action sequence is needed to complete or undo transactions interrupted by system failures. To achieve durable transaction atomicity, the transition to the "committed" state must be accomplished by a single write to non-volatile storage. To these ends Jim Gray defined the Write Ahead Log (WAL) protocol [Gray 1978, Gray 1981a] while at IBM Research. The WAL protocol records the old and new states induced by protected actions separately from the actual state changes. The logged changes are written to stable storage *before* the actual changes are written back to stable storage (that's the "Write Ahead" part). Transactions are committed by simply appending and writing a 'commit' record to the recovery log. Logged changes are used to *undo* protected actions of aborted transactions and of transactions in progress at the time of a system failure. Log records are also used to *redo* committed actions whose actual changes have not been written back to stable storage at the time of a system failure. The WAL protocol allows changed data to be written to their stable storage home at any time after the log records describing the changes have been written into the stable log. This gives the Database Manager great flexibility in managing the contents of its volatile data buffer pools.

The recovery techniques developed by Jim Gray and the System R team have been instrumental to the deployment of on-line transaction processing applications. With the ability to recover from equipment and environmental failures, without loss of committed, protected actions, along with atomic (all-or-nothing) transaction completion, on-line business critical applications become reliable enough to replace batch and paper-based transaction processing. The impact of Dr. Gray's recovery technologies for transaction reliability cannot be overstated – without adequate reliability and durability for transactional applications, the transition to on-line transaction processing would not have been possible.

3. CONCURRENCY CONTROL

In order to facilitate the implementation of correct transaction processing applications, the applications must see a "consistent" database state and transform that state to a new "consistent" state. Early data processing systems ran one transaction at a time, batch style. For performance reasons (either to overlap disk I/O latencies or to exploit multi-processor machines), it is useful to be able to overlap the execution of concurrent transactional applications. If only one transactional application is running at any time, life is simple – only the application logic needs to be correct to ensure continued consistency. Concurrent application execution, on the other hand, needs to isolate each application transaction from seeing or modifying the intermediate states of other uncommitted transactions, while allowing access to its own changes. While at the IBM San Jose Research Laboratory, Jim Gray developed three key ideas related to transaction concurrency control: the notion of transaction serializability; degrees of consistency; and multi-granularity locking.

Jim defined transaction serializability as the ability to re-order the actual action history of concurrently executing transactions to bring together all the actions of each transaction without changing the ordering between read / write or write / write actions on the same item by different transactions [Eswar 1976]. He proved that any serializable action history ensures a consistent final state if each transactional application preserves consistency when run in isolation. Furthermore, he proved that simple read / write locking rules, enforced at the level of the data items, guarantees a serializable action history. Transactions must simply be "well formed" and "two-phase" locked to ensure serializability. "Well formed" transactions lock every item (for read or write) before manipulating the item. "Two-phase" locking requires that no locks be acquired by a transaction once a lock has been released.

Locking, and all other concurrency control methods, ensure serializability by delaying or aborting the progress of the transactional applications. Enforcing serializable execution can induce intolerable delays, deadlocks, and transaction re-tries. Building upon his seminal work in serializability, Dr. Gray invented relaxed locking protocols that sacrifice serializability to reduce concurrency control conflicts while still guaranteeing useful isolation among concurrent applications [Eswar 1976]. Among the relaxed locking protocols he defined are *cursor stability* which enforces repeatable read (completed reads prevent updates by other transactions) only for the current data element of each query result set and *dirty read* which allows reading of uncommitted changes of other transactions. Additional non-serializable locking protocols are possible and several have been incorporated into commercial DBMS products.

Jim Gray not only invented transaction serializability theory and extended it to support relaxed degrees of consistency, he also invented multiple granularity locking protocols that facilitate on-line bulk data activity [Gray 1975]. Multiple granularity locking supports application isolation for not only the finest granularity lockable units (e.g. records) but also for groupings of lower level items into higher level lockable units.

Multi-granularity locking organizes database elements into a hierarchy (actually a lattice) – records are grouped into tables, tables are grouped into table sets, etc. The multi-granularity locking protocol specifies that a lock at any granularity applies to all the elements contained in that granule. Additionally, before acquiring a lock at any

granularity, "intent" locks must be acquired for all higher level granules. Intention lock modes (Intention Exclusive & Intention Share) are compatible with each other and Intention Share is compatible with Read locks. Thus, for example, to Read lock a record, the table set and table containing the record must first be locked Intention Share (in that order). To scan an entire table, the table set is locked Intention Share and the table is locked in Read mode – there is then no need to set Read locks on each record in the table.

The multi-granularity locking protocol enables bulk operations, such as table scan or table delete, without locking every component of the composite object and also facilitates on-line data definition (i.e., DBMS schema changes) by locking high level granules for data definition operations. On-line data definition facilities in the earliest RDBMS prototypes and products were one of the key drivers of the acceptance and success of the Relational Data Model and its early implementations.

3. CONCLUSIONS AND SUMMARY

In summary, Dr. Jim Gray's tenure at the IBM San Jose Research laboratory was spectacularly productive! In the context of the System R relational database research project, Jim developed a model for transactional applications, invented recovery techniques making the database reliable enough to replace paper-based and batch processing record keeping, pioneered serializabilty theory for understanding isolation and consistency issues, introduced relaxed degrees of consistency, and invented multi-granularity locking for bulk operations. Besides making a stunning sequence of technical innovations, Dr. Gray published his results and taught generations of database developers and users all that he had learned and invented [see Gray 1993 for the *full* story]. In addition to making and publicizing his important innovations, he personally implemented (and tested) his algorithms in the System R project (and for commercial products at the IBM Santa Teresa Laboratory).

Finally, I cannot conclude without discussing Jim's collaborative spirit and style. At IBM, Jim worked openly with all the people around him, made them feel part of the process, and accepted and gave constructive criticism on technical matters both great and small. One sign of Jim's collaborative style is the number of authors on the papers that Jim wrote. Working with Jim was a stimulating pleasure as he both challenged his colleagues to participate in the development of his ideas and gratefully accepted their views and participation in refining those ideas. While it was a great loss to IBM when Jim left the company in 1980, we note with pride his seminal contributions to transaction processing technology while at IBM and congratulate him for his continued stream of important accomplishments in the following years.

BIBLIOGRAPHY

[Blas 1981] Mike W. Blasgen, Morton M. Astrahan, Donald D. Chamberlin, Jim Gray, W. Frank King III, Bruce G. Lindsay, Raymond A. Lorie, James W. Mehl, Thomas G. Price, Gianfranco R. Putzolu, Mario Schkolnick, Patricia G. Selinger, Donald R. Slutz, H. Raymond Strong, Irving L. Traiger, Bradford W. Wade, Robert A. Yost, System *R: An Architectural Overview*, **IBM Systems Journal(20)**: 41-62 (1981).

[Cham 1981] Donald D. Chamberlin, Morton M. Astrahan, Mike W. Blasgen, Jim Gray, W. Frank King III, Bruce G. Lindsay, Raymond A. Lorie, James W. Mehl, Thomas G. Price, Gianfranco R. Putzolu, Patricia G. Selinger, Mario Schkolnick, Donald R. Slutz, Irving L. Traiger, Bradford W. Wade, Rogert A. Yost, *A History and Evaluation of System R*, **Comm. ACM 24**(10): 632-646 (1981).

[Eswar 1976] K.P. Eswaran, J.N.Gray, R A. Lorie, I.L. Traiger, *The Notions of Consistency and Predicate Locks in a Database System*, **Comm ACM(19)**: 624-633 (Nov. 1976).

[Gray 1975] Jim Gray, Raymond A. Lorie, Gianfranco R. Putzolu, Irving L. Traiger, *Granularity of Locks in a Large Shared Database*, **Proc.VLDB(1)**, 428-451 (1975).

[Gray 1978] Jim Gray, *Notes on Data Base Operating Systems*, **Lecture Notes in Computer Science(60)**: 393-481 Springer-Verlag (1978).

[Gray 1981a] Jim Gray, Paul R. McJones, Mike W. Blasgen, Bruce G. Lindsay, Raymond A. Lorie, Thomas G. Price, Gianfranco R. Putzolu, Irving L. Traiger, *The Recovery Manger of the System R Database Manager*, **ACM Computer Surveys 13**(2): 223-243 (1981).

[Gray 1981b] Jim Gray, *The Transaction Concept: Virtues and Limitations* (invited paper), **Proc. VLDB(7)**, 144-154 (1981).

[Gray 1993] Jim Gray and Andreas Reuter, *Transaction Processing: Concepts and Techniques*, Morgan Kaufmann, San Mateo, CA (1993).