Jun Rao

450 Computer Science Building
Columbia University
1214 Amsterdam Avenue
New York, NY 10027

Office: (212)939-7054
Home: (212)663-7211
junr@cs.columbia.edu
http://www.cs.columbia.edu/∼junr

Research Interests: Database Systems. I am particularly interested in query processing and cache-conscious algorithms in main-memory database systems, as well as on the optimization and processing of complex queries in decision support and web-based systems.

Education

Ph.D.	Computer Science, **Columbia University** (expected in May, 2000) Advisor: Prof. Kenneth A. Ross	1995-2000
M.Sc.	Computer Science, **Columbia University** **GPA: 4.094/4.0**	1996
B.E.	**Tsinghua University**, Beijing, China major in Computer Science, minor in Applied Mathematics **GPA: 90/100**	1989-1994

Honor

Intern Excellence Award, Sybase IQ Summer 1997

I was the only one among more than 60 interns in the northeastern part of Sybase who was given such an award.

Dissertation: Advanced Query Processing

Complex queries such as correlated queries are common in decision support systems. Traditional nested iteration evaluation method is time-consuming and query rewriting technique does not always apply. I have designed and implemented a new "invariant" technique that can evaluate arbitrary correlated queries efficiently. The technique recognizes the part of the subquery that is uncorrelated and tries to cache and reuse the invariant result. The method can be integrated smoothly into a conventional cost-based optimizer.

As random access memory gets cheaper, it becomes possible to keep database tables memory resident. An important factor in main memory query processing is the cache behavior. Cache miss penalty is high, given the current speed gap between cache access and main memory access. I propose two cache conscious main memory indexing structures: Cache Sensitive Search (CSS) Trees and Cache Sensitive B^+ (CSB^+) Trees. By better utilizing each cache line, both indexes are faster in searching than the existing indexing structures such as B^+-Trees and T-Trees. While CSS-Trees are designed for OLAP environment where updates are batched and infrequent, CSB^+-Trees support incremental updates in a fashion similar to B^+-Trees and can be used for a more general purpose.

Patents

1. "*Cache Sensitive Search (CSS) Tree Indexing System and Method*," Jun Rao and Kenneth A. Ross, filed in 1999 (pending).

2. "*Database System with Methods for Optimizing Performance of Correlated Subqueries by Re-using Invariant Results of Operator Tree*," Roger D. MacNicol, Steve A. Kirk, Randall G. Bello, Jun Rao and Katherine T. Yang, filed in 1998 (pending).

Publications

1. "Making B^+-Trees Cache Conscious in Main Memory," Jun Rao and Kenneth A. Ross, to appear in Proc. of the ACM SIGMOD Conference, 2000.
2. "Cache Conscious Indexing for Decision-Support in Main Memory," Jun Rao and Kenneth A. Ross, in Proc. of the VLDB Conference, 1999.
3. "Reusing Invariants: A New Strategy for Correlated Queries," Jun Rao and Kenneth A. Ross, in Proc. of the ACM SIGMOD Conference, 1998.
4. "Table-based Knowledge Query Language," Kehong Wang, Jun Rao and Lixin Du, in Microcomputer Vol. 12, No.4, China, 1992.
5. "Using EELs, a Practical Approach to Outerjoin and Antijoin Reordering," Jun Rao, Bruce Lindsay, Guy Lohman, Hamid Pirahesh and David Simmen, in preparation, 1999.
6. "Power-pipelining for Enhanced Query Performance," Jun Rao and Kenneth A. Ross, submitted for publication, 1999.
7. "Adapting Materialized Views after Redefinitions: Techniques and a Performance Study," Ashish Gupta, Inderpal S. Mumick, Jun Rao and Kenneth A. Ross, submitted to TODS, 1997.

Research Experience

Research Assistant, Columbia University 1995-present

Query Processing in Main Memory Database Systems 1997-present

I designed and implemented Columbia Main Memory Database System. The goal of the system is to provide fast query performance by making the system cache conscious. The system uses domain and vertical partition to store data efficiently in RAM. The query execution engine uses cache conscious indexing structures for faster query evaluation. The system has more than 10,000 lines of C++ code. I managed a development team of six persons.

Materialized View Adaptation after Redefinition 1996-1997

I investigated techniques for keeping a materialized view up to date when the view definition is changed. I built a system to compare quantitatively the efficiency of adaptation techniques with rematerialization. The system includes an SQL parser, an adaptation analyzer, which can automatically generate the adaptation techniques in SQL, and an interface to a commercial database. The experimental results validate the efficiency of adaptation techniques and show the impact of proper physical designs.

Summer Intern, IBM Almaden Research Center, San Jose, CA Summer 1999

Mentors: Dr. Guy Lohman and Dr. Bruce Lindsay.

I worked on outerjoin and antijoin optimization. Outerjoin and antijoin reordering differs from inner join reordering in that not all the join orders give the correct result. I designed an algorithm for reordering outerjoins and antijoins that can be smoothly incorporated into commercial systems such as DB2. The design distinguishes two approaches: *with compensation* and *without compensation*. The former only considers valid join orders and the latter allows invalid join orders and then compensates the incorrect result. I also did some work on cardinality and cost estimation for outerjoins and antijoins. The cardinality estimation avoids bias and guarantees that different join orders of the same subplan will have the same join cardinality. The work has been prototyped in DB2.

Summer Intern, Microsoft Research Lab, Redmond, WA Summer 1998

Mentor: Dr. Surajit Chaudhuri.

I worked on sampling on queries. As data set gets larger in OLAP systems, query results tend to be large also. Users may not be exactly sure what they want initially. It then makes sense to first provide users with just a random sample of the result. A sample can be obtained much faster if we can avoid materializing the complete result set. The challenge is to reduce the evaluation time and guarantee the randomness of the sample. I designed several techniques for join queries and had them prototyped in Micorsoft's SQL Server system. The result shows that our techniques can give a reasonably good random sample using just a small fraction of the full evaluation time.

Summer Intern, Sybase IQ, Burlington, MA Summer 1997

Mentor: Dr. Roger MacNicol.

I worked on the optimization of correlated queries. The idea is to find the invariant part of the subquery in an execution plan and to avoid the unnecessary reexecution of the invariant part by caching and reusing the invariant result. I found a method of incorporating the invariant technique seamlessly into an existing join optimizer. I finished most of the implementation within three months and the technique now exists in a commercial version of Sybase IQ. In follow-up work, I also investigated the tradeoffs between the invariant technique and the query rewriting technique.

Teaching Experience

Instructor, Columbia University 1997

Taught "Data Structures in C." This is a 3-credit course for non-CS undergraduate students. The class had about 40 students.

Teaching Assistant, Columbia University 1997

Course: Data Structures and Algorithms

Teaching Assistant, Columbia University 1996

Course: Database Systems

Work Experience

Software Engineer, Hua Yi Software Company, Beijing, China 1994–1995

I participated in the implementation of a MIS system for an Import/Export company using Sybase and PowerBuilder over a Novell network. I also designed and implemented an automatic report generating system. Additionally, I was the database administrator of our system.

Professional Activities

Referee for the ACM SIGMOD and VLDB conferences, the ACM TODS journal, the VLDB Journal and the International Journal on Cooperative Information Systems (IJCIS).

Departmental Service

Ph.D. Admission Committee Member 1997, 1998

Implemented a web interface to facilitate the process of applications to the Ph.D. program of the Computer Science Department.