On Linear and Quasi-Linear CPU Times for Most Relational Calculus and Data Mining Queries

Dan E. Willard

University at Albany – SUNY

Alternate Title of Talk: " A **15-minute Summary** of 35 Years of Prior Research PLUS A **5-Minute Look** into Next 20 Years"

Goals of Talk

1. Summarize Content of Our JCSS 1996 and 2002 papers
2. Explain why these results are especially germane today (10 years later)

   Because computer memory costs are roughly 100 times less expensive than they were in the 1996-2002 period

3. Explain Open Questions Raised by These Two Papers for Future Research
4. Explain How these Open Questions Are Germane to Parameterized Complexity Theory

   E.g. The constant inside our $(I \ Log^d I \ + \ U)$ *Times depends on structure of the query q.*

1. Lengthy Literature about Database Optimization

1. C. Beeri, R. Fagin, D. Maier, A. Mendelzon, J. Ullman and M. Yannakakis, Properties of acyclic database schemes, *STOC-1981*
2. C. Beeri, R. Fagin, D. Maier and M. Yannakakis, On the desirability of acyclic database schemes, *JACM* 30(1983) 479-513.
3. P. Bernstein and N. Goodman, The power of natural semijoins, *SIAM J. Comp.* 10(1981), 751-771.
4. D. Goyal and R. Paige, The formal speedup of the linear time fragment of Willard's relational calculus subset, in *Algorithmic Languages and Calculi,* (edited by Bird and Meertens), Chapman-Hall, 1997, 382-414.
5. R. Paige, Formal Differentiation - A Program Synthesis Technique, UMI Research Press, 1981 (277 pp); revised Ph.D. Thesis, NYU, June 1979.
6. R. Paige, Applications of finite differencing to database integrity control and query/transaction optimization, in *Advances in Database Theory,* Vol 2, 171-210, eds. Gallaire, H, Minker, J., and Nicolas, J.M., Plenum Press.
7. C. Papadimitriou and M. Yannakakis, On the complexity of database queries, *J. Comput. System Sci.* 58 (1999) 407-427.
8. Y. Sagiv and O. Shmeuli, Solving queries by tree projections, *ACM TODS* 18(1993) 487-511
9. M. Vardi, The complexity of relational query languages, *STOC-1982*
10. M. Vardi, On the complexity of bounded-variable queries, *STOC-1995*
11. M. Yannakakis, Algorithms for acyclic database schemes, in *VLDB-1981*

Willard differed from Preceding Literature by seeking to combine Advanced Data Structure Indexing Techniques (from Range Query Theory) with Acyclic Optimization Methodologies

2. Ph. D Topic 1975-1978

Willard (1975-1978) looking for a Pragmatic PhD topic

Thesis Adviser: Ugo Gagliardi (supported by ONR) suggests

Relational Database Optimization Because:

1. It is a Quite Unexplored Field during 1975-1978
2. Topic Should Grow Sharply in Importance if Computing Hardware Costs Continue to Drop.

Today's Theme: Year-2013 Ideal Time for Revisiting Topic our JCSS 1996 and 2002 Results:

This is because Computing Costs Have Dropped By Another Factor 100 since the 1996-2002 period when their Results were first introduced.

3. Four Topics in Ph D Thesis

1. General Data Structuring Techniques (Applicable especially to Computational Geometry)
2. Dynamic On-line "E-8" Database Queries
3. Relational Algebra Join Operations
4. Application of Topic 3 to Relational Calculus Queries.

Publications Germane to Topic 1

- JACM Paper with Lueker (1985)
- Willard's SiCompJ (1985) and JACM (1987) improving above result.

Willard initially reluctant to spend much effort on Topics 2-4

> Was it absurdly optimistic (?) to presume that
> Moore's Law's Doubling in Computer Memory Sizes
> would continue for next 35 years (e.g. 1978-2013) ?

Our Pods-1990 and JCSS (1996 & 2002) papers expanded upon Topics 2-4 because Moore's Exponential Growth out-performed ALMOST-EVERYONE'S expectations and made Topics 2-4 viable

4. Notation Used by E-8 Queries

Definition. Given two input variable, x and y, an expression $e(x, y)$ is an E-8 formula iff it combines the following four types of atoms in an arbitrary manner with AND, OR and NOT connectives:

1. Equality Atoms as in : $x.A = y.B$
2. Order Atoms as in: $x.A > y.B$
3. List Atoms as in $x \in L$ where L a pre-specified list.
4. Tabular Atoms as in $(x, y) \in T$ where T a pre-specified table.

Example. $x.A_1 > y.B_1 \wedge \{ x.A_2 > y.B_2 \vee \neg (x, y) \in T_1 \}$

Theorem 1. Let $e(x, y)$ denote an arbitrary E-8 expression, **Y** denote an arbitrary set of y-records and let **I** denote the sum of the "input" cardinalities of **Y** and the "input" lists and tables for L_1, L_2, ... L_j and T_1, T_2, ... T_j. Let **Output**($\bar{x}$) denote:

$$\{ \; y \in \mathbf{Y} \textbf{ where } e(\bar{x}, y) \textbf{ is true} \; \}$$

Then for any **Y** and $e(x, y)$, there exists a data structure $D_e(\mathbf{Y})$ and a constant d where **Output**($\bar{x}$) can be built in time $\text{Log}^d(\mathbf{I})$+OutputSize and which has an $\text{Log}^d(\mathbf{I})$ time for insertions and deletions.

5. Virtues and Drawbacks of Theorem 1

Theorem 1. *(characterizing "dynamic on-line queries)* Let $e(x, y)$ denote an arbitrary E-8 expression, **Y** denote an arbitrary set of y-records and let **I** denote the sum of the "input" cardinalities of **Y** and the "input" lists and tables for L_1, L_2, ... L_j and T_1, T_2, ... T_j. Let **Output**($\bar{x}$) denote:

$$\{ \quad y \in \mathbf{Y} \textbf{ where } e(\ \bar{x}\ ,\ y\ ) \textbf{ is true} \quad \}$$

Then for any **Y** and $e(x, y)$, there exists a data structure $D_e(\mathbf{Y})$ and a constant d where **Output**($\bar{x}$) can be built in time $\text{Log}^d(\ \mathbf{I})$+OutputSize and which has an $\text{Log}^d(\ \mathbf{I}\ )$ time for insertions and deletions.

Difficulties with Theorem 1.

Requires a Separate data structure for each E-8 formula

Virtues of Theorem 1.

1. Possible practical uses in context of modern inexpensive memories
2. "OutputSize" can be omitted from costs for aggregation queries.
3. Modifications of Theorem 1's Procedure for Joins and Aggregate-Joins (on next slide) are **REALLY PRACTICAL ....**

6. A Very Practical Result (in JCSS 1996)

Theorem 2. *(characterizing "join" queries)* Let $e(x, y)$ denote an arbitrary E-8 expression, and **X** and **Y** define the domain-sets over which the variables x and y span. Let **I** again denote the sum of the "input" cardinalities of **X** , **Y** and the "input" lists and tables of L_1, L_2, ... L_j and T_1, T_2, ... T_j . Let **Output**(e) denote:

$$\{ \ (x, y) \in \mathbf{X} \times \mathbf{Y} \ \textbf{where } e(\ x\ ,\ y\ ) \ \textbf{is true} \ \}$$

THEN WITHOUT ANY PREPROCESSING, it is possible to perform the above join in time $O\{ \ \mathbf{I} \bullet Log^{d}(\ \mathbf{I}\ ) + \text{OutputSize}(e) \ ]$ where the constant d depends only on the query e.

Corollaries TO THEOREM 2:

1. The constant d will equal 0 or 1 in almost all applications
2. "OutputSize" can be omitted from costs for aggregation queries.

ABOUT THEOREM 2's PRAGMATIC IMPLICATIONS

1. Should take several person-years to encode, but result is practical.
2. Constant in O-notation should usually be good, but it will depend on e's structural complexity.

7. Notation from JCSS 2002 and its Main Theorem

1. Variables u_i and v_i Range over set R_i and S_i
2. Quantifier " $Q(\ v_i\ )$ " is either Universal of Existential Quantifier
3. Relational Calculus Query looks like
   $\text{Find}(\ u_1, u_2 .... u_k\ )\ \ Q_1(v_1)\,,\ Q_2(v_2)\,,\ ... Q_m(v_m)\ \ :\ \ e(\ u_1, ... u_k, v_1 ... v_m\ )$
4. "Graph" of above query has an arc from Node y to x if some atom in e uses both these variables and x is quantified to left of y

Definition 2. **"RCS""** is the subset of relational calc queries whose graphs are trees or forests with all leaves pointing to roots

Theorem 3 (main result from JCSS 2002) Every RCS query " q ", involving an input of size I , can be executed in time $O\{\ \mathbf{I}\ \bullet\ Log^d(\ \mathbf{I}\ ) + \text{OutputSize}(e)\ ]\}$ where ONLY the constant d depends on the query e **(and no preprocessing is needed)**.

OPEN QUESTIONS ABOUT THEOREM 3:

1. How does the constant inside O-notation depend on query q ?
2. More precisely, we know the constant is often decent, but one should use parameterized complexity theory to better understand it.
3. When can a non-RCS query be translated into an equivalent RCS form, similar to examples in JCSS 2002 ?

8. Surprising Aspects about Theorem 3 and Definition 2

Definition 2. **"RCS""** is the subset of relational calc queries whose graphs are trees or forests with all leaves pointing to roots

Theorem 3 (main result from JCSS 2002) Every RCS query " q ", involving an input of size I, can be executed in time $O\{ \mathbf{I} \bullet Log^{d}(\mathbf{I})+\text{OutputSize}(e)]\}$ where ONLY the constant d depends on the query e **(and no preprocessing is needed)**.

Surprising Fact about Theorem 3 and Definition 4

Many relational calc queries would not satisfy RCS condition iff Tabular Atoms WERE OMITTED from formalism.

But PRESENCE OF THIS PRIMITIVE makes many queries either RCS or convertible to RCS for efficiency purposes

ADDITIONAL QUESTIONS ABOUT THEOREM 3:

1. Are we correct in conjecturing that our formalism, with above caveat, can handle many input queries?
2. More details about how q affects constant in O-notation needed.
3. How should above be implemented in context of low-cost memories?

9. Moore's Law Astonishing Implications for RCS Project

Doubling Computer Memory Size Every 2 Years Entails Factor 250,000 Increase since 1977.

1. Results that seemed wildly theoretical and dizzying in 1977 are Pragmatic today — MUCH TO EVERYONE"S AMAZEMENT.
2. Even results from 1996-2002 have Different Implications Than Before.

This has stunning Implications for Database Optimization, Data Mining and Parameterized Complexity Theory

DID ANYONE ANTICIPATE THIS?

An Amusing Quote from Bill Gates (1984) that Adds Perspective to this Question

> *" Now that a P.C. does have 256K memory capacity, we will NEVER NEED more ..."*

10. Cost-Effectiveness of RCS and Future Work

Examples of Prior Implementations of RCS

1. Robert Paige (in late 1990's) investigated this topic before tragic illness. Wrote with Deepak Goyal (his Ph.D student):

   > "The formal speedup of the linear time fragment of Willard's relational calculus subset, in *Algorithmic Languages and Calculi,* Chapman-Hall, 1997, 382-414.

2. Annie Liu Yanhong supervised G. Priyalakshmi's M.S. thesis
   "Efficient Programs for Solving Relational Calculus Queries"

More Should Be Done in Future:

- Willard (since 2001) focused mostly on Generalizations and Boundary-Case Exceptions for Gödel's Second Incompleteness Theorem in six papers published in JSL and APAL.

  > Topic explain how humans (and futuristic computers) can maintain some approximate *instinctive* knowledge of their own self-consistency ... despite Gödel's barriers ...

- Above is done & W– plans to look at RCS implementations.

Main Point: RCS cost-effective because of **New Memory Costs**